DOID 325: Thinking with Models

Spring 2016, TBA

Teaching Notes (and Class Record)

Steven O. Kimbrough

$Id: TwM-class-s2016-teaching-notes-master.tex 4950 2015-09-13 18:31:27Z sok $

Contents

List of Figures

List of Tables

Preface

Chapter 1

Introduction

1.1 What the Course Is About

Welcome!

1.1.1 Goals for the course

1. Develop in the students facility for engaging in model-based reasoning, whether in science or policy making or other applications.
2. Develop in the students facility for formulating models, whether in science or policy making or other applications.
3. Introduce students to model implementation.
4. Develop in the students facility for thinking critically about models, whether in science or policy making or other applications.

1.1.2 Objectives for the course (in service to the goals)

1. Provide a general survey of models and modeling, whether in science or policy making or other applications.
2. Teach NetLogo as a modeling environment, especially for agent-based models.
3. Teach NetLogo programming, both for developing models in NetLogo and for a gentle introduction to programming for modeling and analysis of models.

4. Support hands-on learning leading to design, implementation, and analysis of models in NetLogo.

1.1.3 Focal Topics

1. Formulating models.

   This comes with practice and is taught by examples and exercises.

2. Agent-based modeling.

   We will focus for the most part on agent-based models. These, more than other types of models are perhaps the easiest for beginners to think about and create. They are a new and rapidly developing style of modeling, in which models are articulated in terms of interactions among individual agents.

3. NetLogo.

   NetLogo is a "low floor, high ceiling" development environment for agent-based modeling. It is inspired by the programming language Logo, which was originally created for the purpose of teaching children how to program. The NetLogo language remains easy for and accessible to beginners, and can be used for quite substantive modeling efforts.

   We will learn to program and develop models in NetLogo. The skills we acquire in doing so are largely transferable to other contexts.

4. Using models in deliberation.

   Model formulation and model-based deliberation are the principal facets of "thinking with models." Here we shall emphasize the principle of solution pluralism as a means for supporting model-based deliberation. As it happens, NetLogo has convenient facilities for this.

1.1.4 Structure of the course

1. Will follow to a large extent the "SAIL" methods and philosophy: Structured, Active, In-class Learning.

   Reading assignments should be done before class. Most classes will be mostly given over to doing in-class exercises. These are required. Some questions will be fairly easy and you should be able to do them if you pay attention in the short lectures and generally keep up with

readings. Bs. Then there will be other questions that you should be able to get if you prepare for class (we'll have very definite material and instructions). As. During the class exercises, we will be here and available to answer questions and help you out. The spirit of the class sessions will be one of collaboration for the sake of learning.

Grading: In-class exercises, plus the term project, and class participation. Maybe other assignments???

2. First ≈2/3 of course will focus on getting up to speed in NetLogo. Then we'll do more survey stuff as you, in parallel, plan and implement your term projects, your models.

Go over syllabus.

1.2 Modeling Examples

1.2.1 WWI Fisheries data and the Volterra Principle; Scale Invasion

Read: **?**, chapters 1 and 2.

1.2.2 San Franciso Bay model

Read: **?**, chapters 1 and 2. See on Youtube:

- `https://www.youtube.com/watch?v=GjTA8Wxi1XA` (2:11 minutes, good)
- `https://www.youtube.com/watch?v=zcetEPDEOFQ` (4:24 minutes, good)
- "Model Saves SF Bay from Ecological Disaster " from *Wired*. Good background info, 3:51. `https://www.youtube.com/watch?v=MiL_TvY1a9g`

1.2.3 Segregation

Schelling model, in NetLogo; see Models Library, Sample Models, Social Science.

1.2.4 Traffic Basic

In NetLogo; see Models Library, Sample Models, Social Science.

1.2.5 Castello Ursino Catania

See "Agent-Based Simulation of Pedestrian Behaviour in Closed Spaces: A Museum Case Study ," by Alessandro Pluchino, Cesare Garofalo, Giuseppe Inturri, Andrea Rapisarda and Matteo Ignaccolo (2014), `http://jasss.soc.surrey.ac.uk/17/1/16.html`, (**?**).

See Figures 1.1 and 1.2, following. Their NetLogo model is [will be] posted on Canvas.

1.2.6 Knapsack

Example of a *constrained optimization model* (COModel).

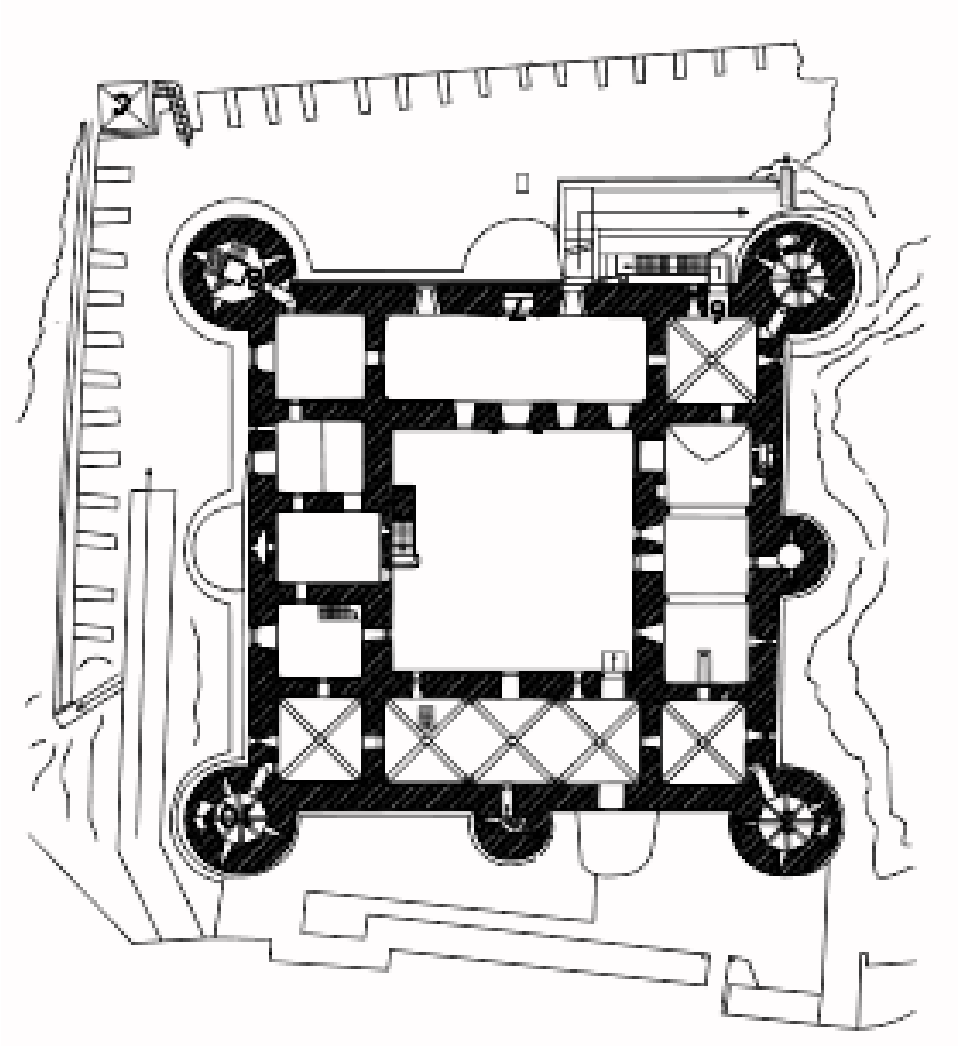

Figure 1.1: Castello Ursino Catania: The castle, now museum, ground floor plan.

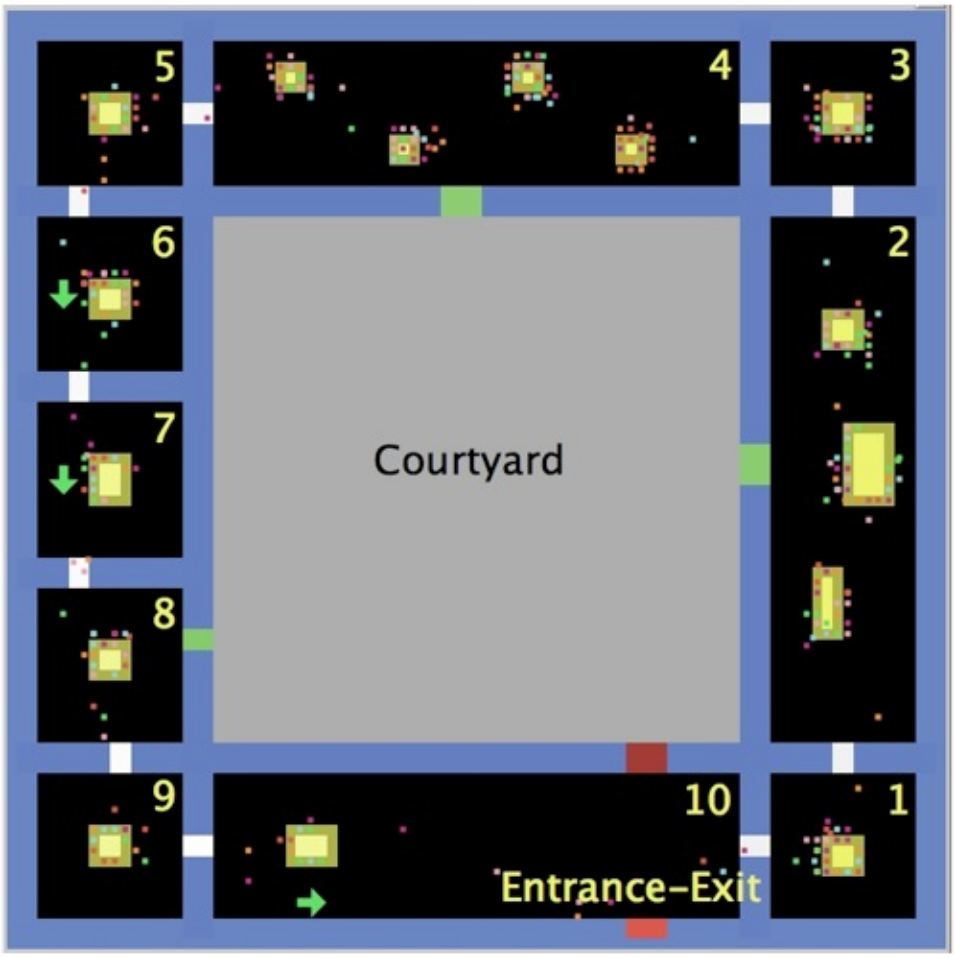

Figure 1.2: Castello Ursino Catania: The NetLogo model user interface.

1.3 Thinking about Models: Some Basic Distinctions

Note that these models are apt for, can be used for, different purposes, e.g., to support a decision (Bay Model), to explain a puzzling finding (LV), to ask "How possibly?" questions (Schelling).

Some distinctions:

1. "Three Kinds of Models" (**?**, chapter 2): concrete models (e.g., SF Bay model), mathematical models (e.g., Lotka-Volterra model, knapsack model), and computational models (e.g., Segregation, Traffic Basic).

   Focus of the course is on computational models and specifically on agent-based models (ABMs, aka: individual-based models or IBMs). But much of what we will learn will apply to other kinds of models.

2. Parametric versus strategic decisions (and models).

   Parametric: decision maker (agent) makes a choice, Nature acts, possibly randomly, and the decision maker is rewarded (positively or negatively). Example: San Francisco Bay model, knapsack model.

   Strategic: decision maker (agent) makes a choice, Nature acts, possibly randomly, and the decision maker is rewarded (positively or negatively), depending in part on decisions made by other agents/decision makers. Lotka-Volterra. Segregation. Traffic Basic. Castello Ursino Catania.

   In this course we'll look at both kinds of decisions and models of them.

3. Insight versus decision models.

   Decision: San Francisco Bay model, knapsack models, and (to a degree) Lotka-Volterra.

   Insight (but not decision): Segregation, Traffic Basic.

   But: Lotka-Volterra, and Castello Ursino Catania.

4. Descriptive versus exploratory models.

   A distinction developed by Steven Bankes and others. See (**??**) for starters.

1.4 Agent-Based Modeling

(**???????**)

1.5 Solution Pluralism: Key to Thinking with Models

our present project, employs in an essential way the perspective of what we call *solution pluralism* (**??**). It will be helpful for framing our project to describe briefly this conceptual point of departure. So, a word about that now.

Solution pluralism in the context of modeling is the principle of obtaining multiple solutions for a model and using the resulting *corpus* of solutions to inform deliberation and decision making. The principle appears elsewhere as well, notably in observation and mesurement, where it is standard practice to obtain a sample of (often, ideally independent) observations or measurements, and then to reason from the sample. This is pervasive and required in most cases; it is what we doing in making survey samples and in weighing small quantities multiple times.

In the context of modeling, the principle of solution pluralism is often realized in sensitivity anlysis, where multiple solutions of a model type are obtained by varying one or more parameters in an instance of the model type. Similarly, when constrained optimization is involved it is standard to obtain multiple solutions to a model for the sake of estimating the Pareto frontier of the decision space, as one or more parameters vary. Corpora of solutions to model instances are also standardly obtained when the models have stochastic elements, as in Monte Carlo and discrete event simulations.

1.6 NetLogo

Ten minute overview.

1.7 For Next Time

1. Install NetLogo on your computer.
2. Bring your computer to class.
3. See the syllabus and do the assigned readings. In particular, read (**?**, chapter 2, pages 45–68) and read in the *NetLogo User Manual* (`http://ccl.northwestern.edu/netlogo/docs/` and installed on your computer with the NetLogo distribution):
   - Tutorial #1: Models
   - Tutorial #2: Commands

- Tutorial #3: Procedures

Also, we'll be reading these, so you may as well get started:

- Interface Guide
- Info Tab Guide
- Programming Guide

See Figure 1.3.

Figure 1.3: NetLogo User Manual.

1.8 For More Information

On agent-based modeling: (**???????**)

Part I

Programming NetLogo

Chapter 2

Working with Patches

2.0 Assigned Readings for the Class Today

1. **?**, chapter 2, pages 45–68
2. *NetLogo User Manual* (`http://ccl.northwestern.edu/netlogo/docs/` and installed on your computer with the NetLogo distribution):

   - Tutorial #1: Models
   - Tutorial #2: Commands
   - Tutorial #3: Procedures

   And the NetLogo program Life Simple, found in the Models Library, *IAMB Textbook* folder, in the *chapter 2* folder.

2.1 Crib Sheet

2.1.1 Navigating NetLogo

1. Three tabs: Interface, Info, Code.
2. In the Code tab, our first *procedure*, a *command procedure*.

   ```
   to hello-world
     print "Hello, world!"
   end
   ```

   Points arising:

(a) General pattern for a command procedure in NetLogo:

```
to <command name>
     <Block of one or more lines of code.>
end
```

(b) `< ... >` is markup indicating that you the programmer need to substitute in appropriate NetLogo expressions or lines of code.

(c) Sequences of characters—called *strings*—are indicated by opening and closing double quotes.

(d) `print <expression>` prints the expression to the the Command Center window on the Interface tab.

3. NetLogo programs are built with procedures. There are two kinds: command procedures and reporters. Both are put into the Code tab, which is a text editor aware of the NetLogo programming language. We'll discuss reporters later.

4. Normally, NetLogo programs will have a procedure called `setup` and a procedure called `go`. This is conventional and appropriate but hardly required.

5. User interface widgets: creating with point-and-click.

   Create a button that *calls* the `hello-world` procedure. Click on it. What happens?

6. Menu items and help.

7. Inspecting patches.

   See Figure 2.1. What happens if you edit the properties?

8. Inspecting turtles.

   See Figure 2.2. What happens if you edit the properties?

9. NetLogo is *not* case sensitive.

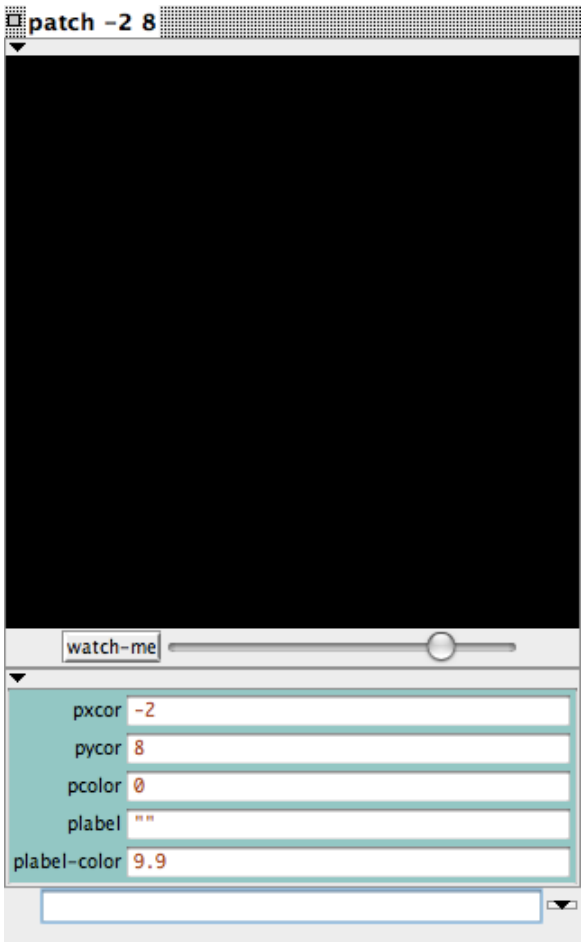

Figure 2.1: Inspecting a patch with default settings.

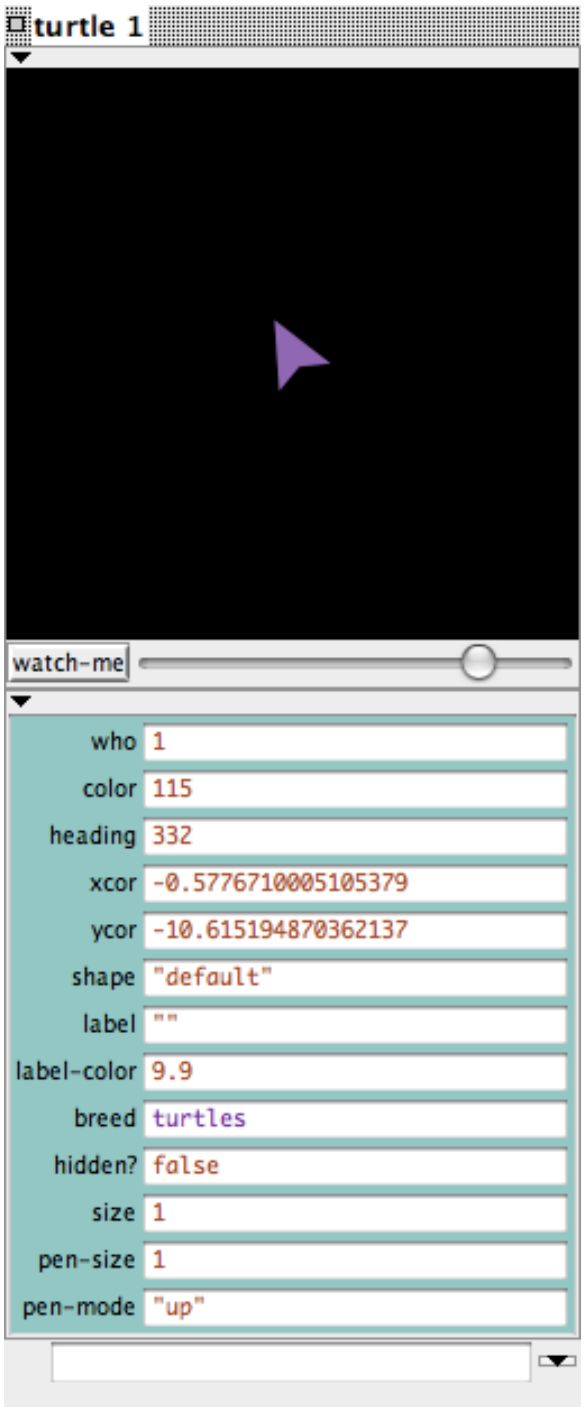

Figure 2.2: Inspecting a turtle with default settings.

2.1.2 The Game of Life

2.1.3 Coding points

To begin, see the readings for today in our textbook: **?**, chapter 2, pages 45–68. We'll begin with the code on page 51.

1. Here is the `setup` command procedure in the Life Simple model:

```
to setup
  clear-all
  ask patches
  ;; create approximately 10% alive patches
    [
      set pcolor blue - 3 ;; dark blue cells are dead
      if random 100 < 10
      [ set pcolor green ] ;; green cells are alive
    ]
    reset-ticks
end
```

   Notice that it fits the command procedure pattern discussed above. Now, let's look into it.

2. `clear-all` is normally the first line of the `setup` procedure. It resets/clears everything that can be reset/cleared.

3. `ask patches [<Block of one or more lines of code.>]`

   Creates a patch context inside the square brackets, randomly orders the patches and then sequentially executes the
   `<Block of one or more lines of code.>`
   on each patch.

4. `ask` is a very important command in NetLogo. See the "ask" section in the NetLogo "Programming Guide". It's what you use to get agents to do things. Patches are agents in NetLogo. So are turtles and links. All in due time.

5. The square brackets after `ask patches`, `[ ... ]`, are necessary, but their placement is quite flexible. Put them as you will for the sake of clarity. The book's example is a good one.

6. `ask patches` is said to create a *patch context* within the square brackets. In this context, commands are automatically addressed to patches. You can think of it this way. When `ask patches` is executed, NetLogo makes a randomized list of patches. It does this each time `ask patches` is executed, so that it processes the commands on the patches in a different, random order each time. Then, having created the randomized list, NetLogo processes the commands on one patch at a time, using the order indicated in the randomized list.

7. `set pcolor blue` This is a deceptively simple command. Understand how it works and you will understand a lot about how NetLogo works. `pcolor` is a patch attribute or property; it comes with every individual patch. See Figure 2.1.

8. `blue` is a NetLogo built-in constant, used for indicating a certain shade of . . . blue! Try typing `blue` Command Center (at the `observer>` prompt. What do you get?

9. `set` is NetLogo's command for assigning a value to a variable. A variable is a symbol whose value can vary, can be changed, so `pcolor` is a variable that attaches to each patch separately. The patten is:

   ```
   set <variable> <value>
   ```

   with the meaning "Give the variable `<variable>` the value `<value>`." In other languages this is typically done with something like:

   ```
   <variable> = <value>
   ```

   but not in NetLogo.

10. Now an important question: When NetLogo executes `set pcolor blue` in the above `setup` procedure, how does it know which patch to set to blue?

    Answer: See above for how `ask` works. NetLogo first makes a random list of the patches, then it works through them one at a time in order. At any given time until the `ask` task completes NetLogo "knows" which patch it is attending to and it executes the commands on that one patch.

11. Execute this procedure a few times to see for yourself:

    ```
    to test
      ca
    ```

```
  ask patches
  [
    set pcolor red
    wait 0.5
  ]
end
```

(Use Halt under the Tools menu to stop execution of the procedure.)

12. Look up and read about the `set` command in the "Programming Guide."

13. Next we have a *flow control* statement, here an `if`.

```
if random 100 < 10
[ set pcolor green ] ;; green cells are alive
```

14. /* here */

2.2 For More Information

/* References on the Game of Life. */

Chapter 3

Working with Turtles

3.0 Assigned Readings for the Class Today

1. **?**, chapter 2, pages 68–87
2. *NetLogo User Manual* (`http://ccl.northwestern.edu/netlogo/docs/` and installed on your computer with the NetLogo distribution):
   - Reference: Interface Guide
   - Reverence: Programming Guide
     - Agents
     - Procedures
     - Variables
     - Tick counter
3. And review for mastery: *NetLogo User Manual* (`http://ccl.northwestern.edu/netlogo/docs/` and installed on your computer with the NetLogo distribution):
   - Tutorial #1: Models
   - Tutorial #2: Commands
   - Tutorial #3: Procedures

3.0 Crib Sheet

`$Id: intro-turtles.tex 4950 2015-09-13 18:31:27Z sok $`

Chapter 4

Working with Plotting

4.0 Readings for the Class

1. **?**, chapter 2, pages 87–99
2. *NetLogo User Manual*: Reverence: Programming Guide
    - Colors
    - Ask
    - Agentsets
    - Breeds
    - Buttons
    - Plotting

4.1 Crib Sheet

$Id: intro-plotting.tex 4950 2015-09-13 18:31:27Z sok $

Part II

Modeling Examples

Chapter 5

Converse and Model Analysis

5.1 The Computational Problem Solving Cycle

Business analytics is about using data and models to solve—or at least to contribute towards solving—decision problems faced by individuals and organizations of all sorts. These include commercial and non-profit ventures, LLCs, privately-held firms, ESOPs, cooperatives, governmental organizations, NGOs, and even quangos. Business analytics is, above all, about "thinking with models and data" (including text data); it is about using them as inputs to deliberative processes that typically are embedded in a rich context of application, which itself also provides inputs to the decision maker.

Focusing on the general analytics at the expense of details of the governing context of any particular case, there are three kinds of knowledge important for our subject.

1. Encoding.

   How to express an algorithm or procedure in the computational environment of our choice and get it to run and produce useful results?

   Important modern computational environments include Excel (and other spreadsheet programs), MATLAB, Mathematica, Python, Ruby, PHP, JavaScript, R, SAS, SPSS, NetLogo, and much else, including core programming languages such as C, C++, and Java.

   We shall discuss coding in this book, but for the most part relegate it to later chapters, aimed at more advanced users, or at least users who would be more advanced and so wish to study programming for business analytics. Python and NetLogo will serve as our focal core

programming languages, although we shall have many occasions to mention others. We will advert to Excel, GAMS, and OPL often, focusing on NetLogo and Excel in the early parts of the book.

2. Solution design.

   Solution design is an issue that arises prior to encoding. Given a problem, how should it be represented so that it can be coded in one's computational environment of choice (NetLogo, Python, Excel, etc.)? Alternatively put: How should we design the representations—the data structures and algorithms—for solving the problem?

   Solution design, including representation, efforts occur at a level of abstraction removed from encoding efforts. The two are not entirely independent, since felicitous choice of computational environment aids and abets representation enormously. Still, good designs will ideally be accessible to, and useful for, all parties in an analytics effort, programmers and non-programmers alike.

3. Analytics: Post-solution analysis.

   Given a solution design, its encoding into a model, and successful execution of the model, we arrive at a solution to the model. Now the real work begins! We need to undertake *post-solution anlaysis* (also known as *model analysis*) in order to validate the model, test and understand its performance, obtain information perhaps relevant to reconsidering the assumptions of the model, and so on.

   Further, how can we use data, text, and models to solve business problems (broadly conceived)? Or given a business problem, how shall we approach its solution with data, text, and models (assuming it is amenable to such)? In short, how can and how should we go about "thinking with data and models" in the context of real problems?

   Post-solution analysis (or model analysis) refers, then, to that variety of activities we undertake after a model has been designed, formulated, and solved. These activities may be directed primarily at testing and validating a model or at supporting deliberative decision making with the model, although in fact it is often pointless to maintain a distinction between the two goals. In any event, the activities characteristic of post-solution analysis are typically useful and used both for validating a model and for deliberating with it.

The main focus of this book is on how to address these questions with specifics, to provide recommended and usable techniques, along with realistic examples. We emphasize *using* implementations that let us exercise and explore models and data for the sake of discovery, understanding, and decision making. To this end we dwell at length throughout on post-solution analysis. Its techniques and methods are not only accessible to all who would engage in thinking with models, they are also essential to actual deployment and decision making with models.

These levels are interdependent. Coding depends on representation and representation depends on what the business problems are. Conversely, to do analytics we need to specify the problem rigorously, meaning we need to find implementable representations and we need to code them up. And then, we emphasize, the real work begins: we need to use our implementations to explore and discover what our models and data can tell us. To really do analytics we need the results of representation and encoding, but that is just the beginning.

The book is about all three levels. In the beginning, however, and for much of the book we emphasize analytics and de-emphasize representation and encoding, so that readers will have much to sink their teeth into without raising the hackles of those adverse to computer programming.

Let us not forget the larger context of business analytics. We can embed our levels in the larger context of a *computational problem solving cycle.* See Figure 5.1, which we might frame as an updating or complement for the twenty-first century and new technology of Peirce's "The Fixation of Belief" **?**.

The last three steps in the cycle correspond to, or encompass, our three levels. The first step—problem recognition—gets the ball rolling. Throughout, we aim to provide realistic information on representative problem situations. This will serve to motivate the analyses and to expose the reader to real world considerations.

The second step in the cycle framework is to find a solution concept for the recognized problem. By this we mean a general approach that is likely to be successful in addressing an identified aspect of the problem. Should we build a model or take a data-driven approach? If we build a model, what sort of model should we build?

Finally, to turn the list into a cycle, we emphasize that every step taken, every decision made, is provisional. Computational problem solving efforts iterate through the list and revisit steps until deliberation is—provisionally—abandoned in favor of action.

Enough abstraction for the present! We now turn to an example that

touches upon and illustrates these principles.

5.2 Example: Converse's Formula

We now step through the computational problem solving cycle of Figure 5.1, working at a high level, but using a concrete example.

5.2.1 Problem Description

A competitor has a store in city A. There is no other potentially competitive store in the region. If we put a competing store of our own in city B, will it attract enough traffic to be profitable? We will not answer this question completely, only partially. We will construct a relevant model.

5.2.2 Solution Concept

We use Converse's formula (**?**, page 432):

$$D_b = \frac{D_{ab}}{1 + \sqrt{\frac{P_a}{P_b}}} \tag{5.1}$$

where

- D_b: The breaking point between city A and city B in distance from B.
- D_{ab}: The distance separating city A and city B.
- P_b: The population of city B.
- P_a: The population of city A.

This, with data, may give us a rough estimate of the viability of placing a store in city B. If we can then determine how many potential customers reside on the city B side of that breakpoint, we will get an estimate of the size of the *catchment area* ("the area and population from which a city or individual service attracts visitors or customers") and this will be informative regarding whether a store at B will be profitable.

5.2.3 Design

5.2.3.1 Data Structures

Here things are very simple. We plan to implement a simple formula, for which we only need the four scalar values as variables.

5.2.3.2 Algorithm

Again, things are very simple. Our algorithm can be expressed easily in standard computer coding languages, mirroring closely the algebraic expression, Converse's formula, expression (5.1). In fact, we shall implement a slight generalization of it, replacing 2 in $\sqrt{(\bullet)} = (\bullet)^{\frac{1}{2}}$ with r, yielding:

$$D_b = \frac{D_{ab}}{1 + \left(\frac{P_a}{P_b}\right)^{\frac{1}{r}}} \tag{5.2}$$

5.2.4 Encoding

Figure 5.2 shows the user interface of a NetLogo program that implements the Converse formula. The program file is *Converse Formula.nlogo*. It is available at the book's web site and at the NetLogo Modeling Commons `http://www.modelingcommons.org`. To run the program you will need to download and install NetLogo on your computer. The NetLogo home page is `https://ccl.northwestern.edu/netlogo/`. There, you can download NetLogo and access a rich corpus of information about NetLogo.

The four parameters of the model may be set by adjustable sliders at the bottom of the interface window: $D_{ab} \Rightarrow$ `Dab`, $P_a \Rightarrow$ `Pa`, $P_b \Rightarrow$ `Pb`, and $r \Rightarrow$ `root`. As you move them, the value of D_b ($\Rightarrow$ `Db`) is recalculated at displayed in a small labeled box (a NetLogo *monitor*) at the top of the window. In Figure 5.2 its value is 31.55. If you right-click on this monitor a new window will pop up displaying the implementation of the formula. Figure 5.3 shows what you will see.

The three plots in Figure 5.2 let us chart how D_b responds to changes in D_{ab}, *root*, and P_b respectively, taken one at a time.

5.2.5 Employment of the Model

The Converse formula tells us that if cities A and B are 100 kilometers apart, city A has a population of 400,000, and city B has a population of 85,000, then the breakpoint is at about 31.55 kilometers.

Besides implementing the Converse formula and letting us easily and quickly calculate it for different values of the parameters, the model supports us in exploring various *post-solution analysis and exploration* questions, such as:

1. How large would the population of city B need to be in order for the breakpoint to be at least 37.0?

2. How small would the population of city A need to be in order for the breakpoint to be at most 40.0?

3. Given fixed values of the other three parameters and a D_b value that has a margin of safely of 5 kilometers for maintaining profitability, over what range of r do we remain within this margin of safety?

1. Recognition or detection of a problem.

   Describe the problem.

2. (Computational) solution concept.

   Develop an approach, a computational approach, for solving the problem—or at least for providing useful information about it.

3. Solution design.

   A good slogan in this context: "Algorithms + Data Structures = Programs" **?**.

   (a) Data structures

   (b) Algorithms

   The result of this step is typically a representation, perhaps in *pseudocode*, that is amenable to encoding. The representation should be independent, or abstracted from, any particular programming language, although it may be developed with one in mind.

4. Encoding.

   Translate the design into a working computational entity, aka implementation, that produces results (solutions).

5. Post-solution analysis.

   Probe the encoded model in order to measure and assess its performance under a range of test conditions.

Figure 5.1: Computational problem solving cycle.

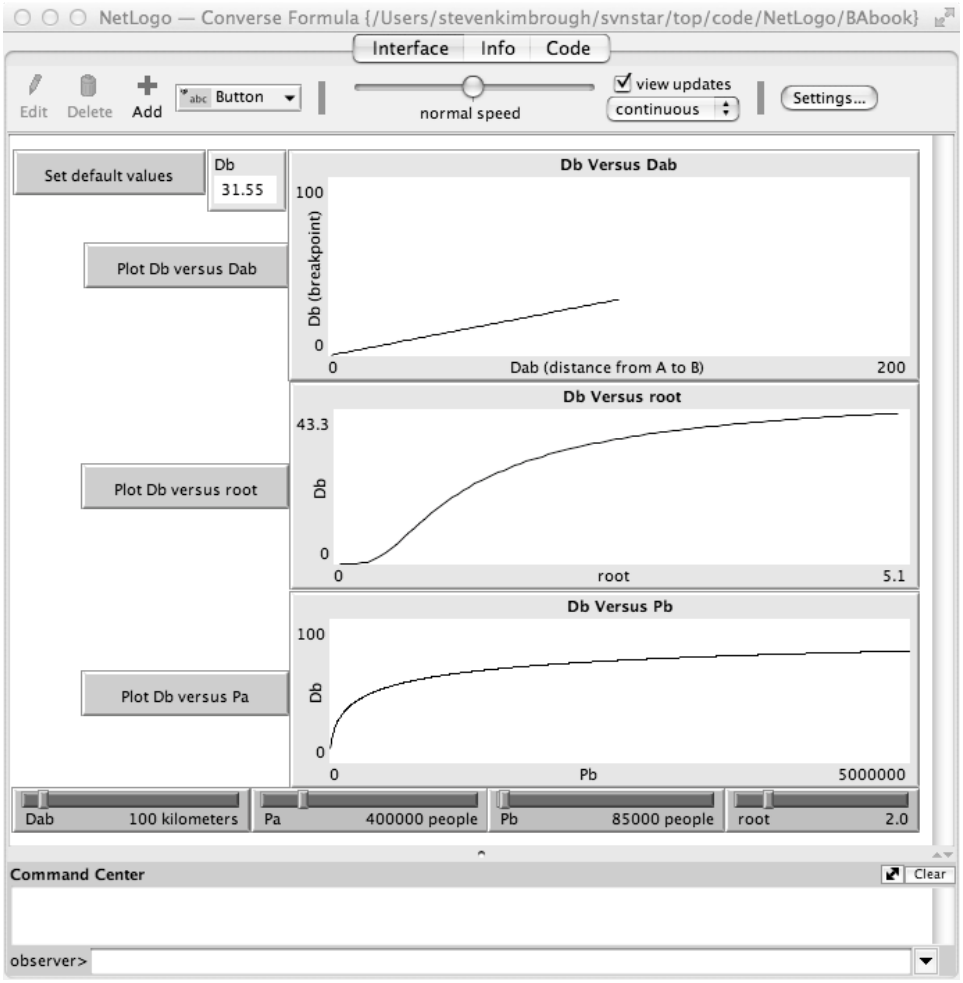

Figure 5.2: Converse Formula model, a NetLogo implementation of Converse's formula.

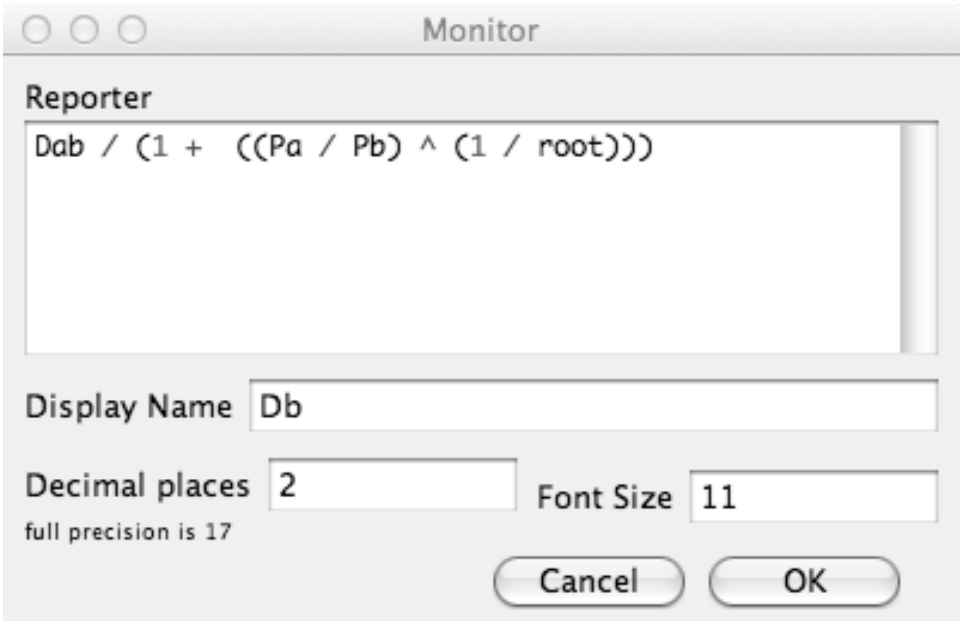

Figure 5.3: NetLogo implementation of the Converse formula expression, in a monitor widget.

5.3 Scoping Out Post-Solution Analysis

Because post-solution analysis figures so importantly in what follows we shall discuss it now in some detail, describing the several facets of the concept. Later chapters will refer to this section and elaborate upon it.

5.3.1 Sensitivity

To begin, it is very often not a good assumption that each of the model's parameters—D_{ab}, P_a, and P_b in the case of Converse's formula—are known with a very high degree of certainty. We can estimate them, with more or less precision, and solve the model on the basis of our best estimates, but when it comes to implementation of any decision, we may find that our best estimates in fact vary from reality. This is most often the case, and if so, then a number of *sensitivity analysis* questions arise, prototypically:

> *Are there small changes in parameter values that will result in large changes in outcomes according to the model? If so, what are they and which parameters have the largest sensitivities, i.e., which parameters if changed by small amounts result in large changes in model outcomes? Which parameters have small sensitivities?*

Points arising:

1. What counts as a large change or a small change to a parameter depends, of course, entirely on the context and the ambient levels of uncertainty. Typically, we might think of changes on the order of a few percent in the level of a parameter as small. Perhaps a better measure is the amount of uncertainty or of variation not subject to our control in the value of a parameter. In the present case, perhaps P_a is only comfortably certain within, say, 5% of its assumed value. If so, we have a natural range of exploration for sensitivity analysis.

2. What counts as a large change or a small change in the outcome(s) of the model depends in large part on the model in question and on the overall context. In an application of the Converse formula we might had a policy constraint to the effect that, say, D_b must be greater than 31 if we are to build store B in the presence of A. We will naturally be interested in whether the results of the model are feasible in this sense and, with sensitivity analysis, whether small changes in the parameter values will make feasible (constraint satisfying) results infeasible

(constraint violating) and vice versa. This will especially be the case with optimization models (our discussion of these begins in chapter **??**), where we will be interested in knowing whether any small change in parameter values will make the model infeasible. That is, if we make some "small" changes (however defined) to the model's parameters and then re-solve the model, could we find that there are no longer *any* feasible solutions?

3. Less drastically, while we might be confident that small changes in the realized parameter values will not lead to infeasibility, there remains the possibility that small changes can drastically degrade the value of the optimal solution based on the estimated parameter values. In the present case, is it possible or likely that a small change to P_a or P_b would greatly change the value of our present solution? Could it happen, we must ask, that the present results to the model would have disastrous consequences if implemented in the presence of very minor parameter value changes?

4. We are also interested in knowing which parameters are comparatively insensitive, so that relatively small changes to them have very little effect on outcomes upon implementing the solution to the model.

5. We will need to consider all of our sensitivity questions both in the context of changes to a single parameter and to the context of simultaneous changes to multiple parameters.

These are among the main general considerations for sensitivity analysis. The literature is extensive and the topic itself open-ended. These remarks indicate the spirit of sensitivity analysis as a means of opening, rather than closing, the discussion of its scope and extent.

5.3.2 Policy

In a second category of post-solution analysis questions, the possibility may arise of altering decision variable levels because of considerations not reflected in the model. This possibility is often present, and if so, then a number of *policy analysis* questions arise, prototypically:

> *Are there policy reasons, not represented in the model, for deviating from either choosing a preferred (e.g., optimal) solution identified by the model or for modifying one or more parameter value? If so, what changes are indicated and what are the consequences of making them?*

> *Policy questions are about factoring into our decision making qualitative factors or aspects of the situation that are not represented in the model formulation.*

In a typical case, it may be desired for business reasons to set a decision variable to at least a minimal level. To take ta hypothetical example, there may be reasons having to do with, say, maintaining supplier relationships, for purchasing at least 70 units of grain 2, instead of the 50 that is optimal in terms of the model. The analysis question is then how best to do this and figuring out how much it will cost. If costs are too high that may well override any policy factors. In this typical mode of decision making, policy issues may be ignored and the model solved, after which an investigation in undertaken to determine whether to make changes in accordance with the policy factors.

What we are calling policy questions frequently raised as what are referred to as *intangibles,* factors that matter but are difficult to quantity, difficult certainly to express adequately as part of a mathematical programming (or other model) formulation. The model, however, can be most helpful for deliberations of this kind. If, to continue the example, there are policy reasons for favoring purchase of at least 70 units of grain 2, then the model can be used to estimate our opportunity cost were we to do this. Generally speaking, having a solution to the model at hand is materially useful for evaluating the cost of taking a different decision. Which decision to take is always up to the decision maker. A model serves its purpose well when it informs the decision maker about the costs and benefits of alternatives.[1]

5.3.3 Outcome Reach

In the present example, the model outcome is $D_b = 31.55$. What if instead of 31 we require at least 33? What combinations of parameter value changes will achieve this? and among them, how likely are they to be realized or achievable with action on the part of the decision maker?

This is an example of an *outcome reach* question on the improvement side. There are degradation side questions as well. For example, anything less than 29 might be disastrous. What combinations of parameter changes would lead to such a result, how likely are they, and what might we do to influence them? Prototypically for this third category of post-solution analysis questions we have:

[1] Thanks to Frederic H. Murphy for emphasizing to us the vital importance for model-based decision making of qualitative, extra-model factors, which we call *policy questions.*

Given an outcome predicted or prescribed by the model, and a specific desired alternative outcome, how do the assumptions of the model need to be changed in order to reach the desired outcome(s)? Is it feasible to reach the outcome(s) desired? If so, what is the cheapest or most effective way to do this? Outcome reach questions arise on the degradation side as well. By opting to accept a degraded outcome we may free up resources that can be used elsewhere.

5.3.4 Opportunity

Looking forward to constrained optimization models (chapter **??** and later), consider properties of a model discussed in chapter **??**. At optimality the constraints on nutrients A and B are *tight.* In our model formulation, Figure **??**, expression (**??**) states the constraint that the mix of feeds must provide at least 1,250 units of nutrient A, and expression (**??**) states the constraint that the mix of feeds must provide at least 250 units of nutrient B. As we shall see, in the optimal solution both of these constraints are satisfied with equality. That is, in the optimal solution we supply exactly 1,250 units of nutrient A and 250 units of nutrient B. This raises the possibility that for a price we might be able to relax one or more of these constraints and thereby get a better result.

Prototypically, and more generally, for this fourth category of post-solution analysis questions we have:

What favorable opportunities are there to take action resulting in changes to the assumptions (e.g., parameter values) of the model leading to improved outcomes (net of costs and benefits)?

These have also been called *candle lighting questions* in allusion to the motto of the Christopher Society, "It is better to light one candle than to curse the darkness" **?????**.

5.3.5 Robustness

Something is robust if it performs well under varying conditions **?**. Wagner's characterization is representative and applies to any system: "A biological system is robust if it continues to function in the face of perturbations" (**?**, page 1), although we will normally want to add "well" after "function". (See also **??**.) This general notion of robustness is apt for, is important for, and is useful in many fields, including biology, engineering,

and management science. The general notion, however, must be operationalized for specific applications. We focus in this book on management science applications, especially applications to optimization.

This fifth category of post-solution analysis questions may be summarized with:

> *Which solutions or policy options of the model perform comparatively well across the full range of ambient uncertainty for the model?*

How exactly we operationalize the general concept of robustness, in the context of decision making with models, requires extensive discussion which will appear in the sequel as we discuss various examples.

5.3.6 Explanation

Models, especially optimization models, are typically used *prescriptively,* used that is to recommend courses of action. An optimal solution comes with an implicit recommendation that it be implemented. Because it is optimal the burden of proof for what to do shifts to any envisioned alternatives. In fact much of what post-solution analysis is about is deliberating whether to take a course of action other than that given by the optimal solution at hand.

It is natural in this context for any decision maker to request an explanation of *why* what is recommended is recommended. Why X rather than some favored alternative Y?

> *Why X?: Given a predicted or prescribed outcome of the model, X, why does the model favor it? Why not Y?: Given a possible outcome of the model, Y, why does the model not favor it it over X?*

Questions of this sort are quite often encountered in practice and handed with appeal to common sense. For example, constraints might be added to the model to force the outcome Y. This will result either in a degraded objective function value or outright infeasibility. In either case, the model can be helpful in explaining why Y is inferior to to X, particularly by suggesting outcome reach analyses that add insight (e.g., that the price on a certain parameter would have to fall by at least a certain amount).

Although explanation questions are commonly engaged it remains true that their is neither settled doctrine or methodology on how best to undertake explanatory analysis with optimization models, nor is there bespoke software support for it. Practice remains unsystematic in this re-

gard. The pioneering work by Harvey Greenberg in the context of linear programming— **????**—remains the point of contemporary departure.

5.3.7 Resilience

Robustness and resilience are closely related concepts. Neither is precisely defined in ordinary language. More careful usage requires a degree of stipulation—"This is what we shall mean by…"—and many such stipulations have been given, if only implicitly. In short, the terms do not have established standard meanings, and the meanings that have been given are various and mutually at odds. We shall use robustness as described above: something is robust if it works reasonably well under varying conditions. We reserve resilience for a form of robustness achieved by action in response to change. A robust defense resists attack by being strong in many ways, in the face of multiple kinds of aggressive forays. A resilient defense is robust in virtue of an effective response to an attack.

Robustness may be static or dynamic. Resilience is dynamic robustness. A resilient solution is one that affords effective decision making after now-unknown events occur. While it is generally desirable to delay a decision until more information is available, this might not be possible or it may come with a prohibitive cost. In some cases, clever design will permit delayed decision making or put the decision maker in position to adapt cheaply to a number of possible eventualities.

Summarizing, these are prototypical resilience (dynamic robustness) questions.

> *Which decisions associated with the model are the most dynamically robust? That is, which decisions best afford deferral of decision to the future when more is known and better decisions can be made?*

* * *

Figure 5.4, page 44, summarizes our seven categories of post-solution analysis questions. We emphasize that boundaries are fuzzy and urge the reader to attend less to clarifying the boundaries and more to using the framework to suggest useful questions leading to better decisions. We mean our framework to stimulate creative deliberation, rather than to encode settled knowledge. Decision making with any model occurs within a context broader than the model. That context may be simple and straightforward, it may be complex and heavily strategic, involving consideration of possible actions by many other decision makers, and it may be anywhere in between.

1. Sensitivity

   Are there small changes in parameter values that will result in large changes in outcomes according to the model? If so, what are they and which parameters have the largest sensitivities, i.e., which parameters if changed by small amounts result in large changes in model outcomes? Which parameters have small sensitivities?

2. Policy

   Are there policy reasons, not represented in the model, for deviating from either choosing a preferred (e.g., optimal) solution identified by the model or for modifying one or more parameter value? If so, what changes are indicated and what are the consequences of making them? Policy questions are about factoring into our decision making qualitative factors or aspects of the situation that are not represented in the model formulation.

3. Outcome reach

   Given an outcome predicted or prescribed by the model, and a specific desired alternative outcome, how do the assumptions of the model need to be changed in order to reach the desired outcome(s)? Is it feasible to reach the outcome(s) desired? If so, what is the cheapest or most effective way to do this? Outcome reach questions arise on the degradation side as well. By opting to accept a degraded outcome we may free up resources that can be used elsewhere.

4. Opportunity

   What favorable opportunities are there to take action resulting in changes to the assumptions (e.g., parameter values) of the model leading to improved outcomes (net of costs and benefits)?

5. Robustness

   Which solutions or policy options of the model perform comparatively well across the full range of ambient uncertainty for the model?

6. Explanation

   Why X?: Given a predicted or prescribed outcome of the model, X, why does the model favor it? Why not Y?: Given a possible outcome of the model, Y, why does the model not favor it it over X?

7. Resilience

   Which decisions associated with the model are the most dynamically robust? That is which decisions best afford deferral of decision to the future when more is known and better decisions can be made?

Figure 5.4: Seven categories of representative questions addressed during post-solution analysis.

5.4 Parameter Sweeping: A method for post-solution analysis

Briefly put, a parameter sweep of a model is an experiment in which multiple model outcomes are produced by executing an algorithm numerous times that solves the model with different parameter configurations. NetLogo, with its BehaviorSpace feature, emphasizes parameter sweeping as a key part of modeling. This is from the *NetLogo User Manual* :

> BehaviorSpace is a software tool integrated with NetLogo that allows you to perform experiments with models.
>
> BehaviorSpace runs a model many times, systematically varying the model's settings and recording the results of each model run. This process is sometimes called "parameter sweeping". It lets you explore the model's "space" of possible behaviors and determine which combinations of settings cause the behaviors of interest.

Of course we can do parameter sweeping without recourse to NetLogo. To illustrate, the following code implements the Converse formula in MATLAB.

```
function [ Db ] = converse_formula( Dab, Pa, Pb )
%converse_formula Calculates the Converse formula.
% Dab is the distance between establishment a and
% b. Pa is the population of a, Pb is the population
% of b. Db = Dab/(1 + sqrt(Pa/Pb)). Db is the
% predicted distance from b in the direction of a
% before which customers will go to b, after which
% customers will go to a.
Db = Dab / (1 + sqrt(Pa/Pb));
end
```

It is not necessary for present purposes that you understand this code, although we imagine that its meaning is reasonably clear. The important point is that once a model is implemented and executable on a computer (as this MATLAB function is) we can write a program that iteratively feeds it different parameter values and collects the outputs, which are the resulting values of the function.

Specifically, we can set P_a to a sequence of values, say 390,000, 395,000, 400,00 (our original value), 405,000, and 410,000, and we can set P_b to a

	75000	80000	85000	90000	95000
390000	30.4845	31.1727	31.8267	32.45	33.0453
395000	30.3497	31.0362	31.6887	32.3105	32.9046
400000	30.2169	30.9017	31.5527	32.1731	32.7659
405000	30.0861	30.7692	31.4187	32.0377	32.6292
410000	29.9572	30.6387	31.2866	31.9043	32.4945

Table 5.1: Parameter sweep results for the Converse formula. Rows: P_a values. Columns: P_b values.

different sequence of values, say 75,000, 80,000, 85,000 (our original value), 90,000, and 95,000. This yields $5 \times 5 = 25$ distinct parameter combinations for which we can obtain function values by executing the Converse function code. Table 5.1 presents the results of calculating the Converse formula for these 25 combinations of parameter values.

This simple example may serve as a prototype for much that is to come. We will confine ourselves here to just two points.

1. In two dimensions, by sweeping two parameters as in Table 5.1, it becomes natural to think of the ensemble on a geographic analogy. We see in the present case that when P_b is equal to or greater than 85,000, the formula value exceeds 31, regardless of the P_a values present. More precisely, when $P_b \geq$ 85,000 or when P_b = 80,000 and $P_a \leq 395,000$, the function value exceeds 31; otherwise it does not. Thus, with our constraint of 31, discussed above, we see two regions or two *phases* of the parameter space revealed by the sweep.

   This point, that parameter sweeping may reveal regions or phases of interest applies, of course, to ensembles of more than two parameters. And it applies to models of all kinds.

2. Generalizing further, parameter sweeping *when it is practicable* can be used as a basis for answering all or nearly all of our post-solution analysis questions. This is a subject we will develop in depth throughout the book.

5.5 Discussion

We have introduced the computational problem solving cycle and Converse's formula as a very simple example of it, along with an implementation of the formula as a NetLogo model, *Converse Formula.nlogo.*

We chose the Converse formula example for the sake of illustrating the computational problem solving cycle. The example is nonetheless realistic. Businesses have used Converse's formula, with various elaborations, for exactly the purpose shown.

A point of emphasis here and throughout the book is that analytics are not static. To do business analytics we need to exercise implementations in order to explore our models and data. Obtaining a solution—a number from a model, a plot of a body of data, and so on—is hardly enough. Business analytics is above all about *post-solution analysis and exploration.* For this purpose, implementations should be seen as providing tools for exploration. The convenient graphical user interface supplied by the Converse Formula NetLogo should be seen as an elementary example of an *analytics dashboard*. These concepts will be with us throughout as we delve into business analytics.

5.6 For Exploration

1. Critically assess the Converse formula as a model. What are its strengths? limitations? key assumptions? Under what conditions is it likely to be useful? not very useful?

2. In addition to the post-solution analysis questions discussed in §5.2.5 for the Converse formula, what other interesting questions are supported by the Converse Formula NetLogo model? Discuss and explain why they are interesting or likely to be useful. What about interesting questions that are not well supported by the model?

3. The following passage appears at the end of §5.1.

   > . . . we might frame [Figure 5.1] as an updating or complement for the twenty-first century and new technology of Peirce's "The Fixation of Belief" **?**.

   Really? Do you agree? or not? Discuss.

4. What is the epistemological status of Converse's formula? It is evidently not a law of nature, although it is expressed like one. Does it support counterfactuals? Is it true?

5. Reimplement the Converse Formula NetLogo model in a spreadsheet program, such as Excel. Compare and contrast the two implementation environments, NetLogo and spreadsheets, for this purpose.

6. Reimplement the Converse Formula NetLogo model in Python. Compare and contrast the two implementation environments, NetLogo and Python, for this purpose.

7. Assuming we have a model for which extensive parameter sweeping results can be obtained. Discuss how to use parameter sweeping to answer the questions of post-solution analysis, discussed in the chapter.

Chapter 6

Exploring the Huff Model

We began in Chapter **??** what will be a series of studies of business analytics examples by looking at a quite simple model, Converse's formula for locating a single store in the neighborhood of an incumbent establishment. The main themes of that chapter—the computational problem solving cycle, the emphasis on post-solution analysis and exploration, the focus on implementation and an analytics dashboard—will be with us throughout the book. They are vey much present in this chapter, where we focus on the Huff model, which can be seen as in the historical line of development ensuing upon Converse's formula.

Besides the Huff model, we introduce in this chapter two new NetLogo models, the distinction between parametric and strategic decision making, greedy hill climbing, and the concept of agent-based modeling. These concepts will be part of our armamentarium going forward.

6.1 The Huff Model

6.1.1 Problem Description

A number of establishments that compete for regional customer exist or are contemplated in a particular geographic area. Given the several locations and attractivenesses of these establishments, we would like to predict the market share each will achieve. We are interested in deciding whether to build a number of new establishments as well as where to put any we do decide to build.

This problem falls under the category of *market area analysis,* addressing one of the category's main concerns: the location of prospective retail

stores. The Huff model is used extensively for this purpose.

6.1.2 Solution Concept

We use the Huff model as the main basis for our analysis. The Huff model is expressed mathematically as follows.

$$P_{ij} = \frac{\frac{S_j^\alpha}{T_{ij}^\beta}}{\sum_{j=1}^{n} \frac{S_j^\alpha}{T_{ij}^\beta}} \tag{6.1}$$

where

- P_{ij}: The probability of a consumer at a given origin i traveling to a particular shopping center j.
- S_j: The size of a shopping center j (typically measured in terms of the square footage of selling area devoted to the sale of a particular class of goods). More generally, a measure of attractiveness of the shopping center (with higher values more attractive than lower values).
- T_{ij}: The travel time or cost for customer i to go to shopping center j.
- α: A parameter, to be estimated empirically but often simply set to 1, for scaling the importance of the attractiveness measure for the establishment, S_j.
- β: A parameter, to be estimated empirically but often simply set to 2 (hence the literature often speaks of the "Huff gravity model" for this class of models), reflecting the effect of travel time and other costs on various kinds of shopping trips.
- n: The number of establishments or shopping centers under consideration.

Further, it is presumed in using this model that there are m customers, that each customer has a known location, and that each of the establishments under consideration has a known location, real or hypothetical.

6.1.3 Design

The model is quite straightforward. Once we represent and record

1. the locations of the customers who will potentially visit the establishments,

2. the locations (real or hypothesized) of the establishments (such as shopping centers) under consideration, and

3. the attractiveness (e.g., size) scores of the establishments under consideration

it is a simple matter to calculate Euclidean distances as the measure of the transportation costs, the T_{ij}'s. Of course, including more realistic measures of transportation costs is a ask that can be arbitrarily onerous. We take this matter up in the discussion section, §6.2.

An organization deciding whether, and if so where, to locate a number of establishments will not be directly interested in the outputs of the Huff model, the P_{ij}'s. It simply does not matter what the probability is that a particular individual i will shop at a given store, j. What matters is the overall level of customer traffic. This can be estimated with the Huff model by summing the probabilities for each store over the customers to get the expected traffic for each establishment:

$$\text{expected\_traffic}_j = \sum_{i}^{m} P_{ij}, \quad \forall j = 1, 2, \ldots, n \tag{6.2}$$

6.1.4 Design & Encoding 1

Figure 6.1 shows the Interface tab of the Huff NetLogo model. There are 1000 customers in the region, scattered randomly by the setup procedure and displayed as gray dots. There are four establishments present, represented as black squares. The report in the output widget in the lower left-hand corner of the window tells us that the establishment at location 2, -1 has the highest expected traffic, at 362.92.

The number and placement of stores in Huff are determined interactively. After initialization with the setup button, the user clicks the add-drop-stores button and clicks to add or delete a store. When finished adding and deleting stores, the user clicks the add-drop-stores button again and proceeds to evaluate the traffic per store by clicking on the calculate-store-traffic button. Such is the basic drill with the Huff NetLogo model. Many iterations and variations on it are possible for doing interesting analytics.

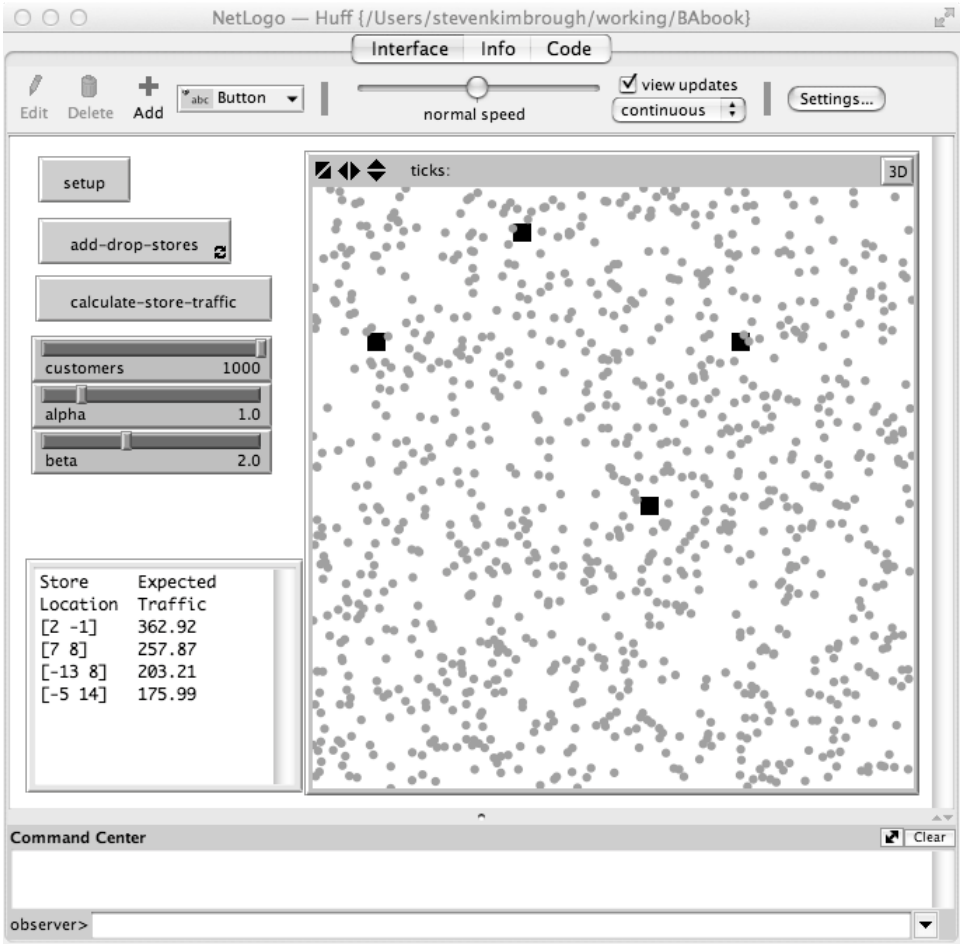

Figure 6.1: Huff, a NetLogo implementation of Huff's model.

Most significantly regarding our design and encoding of the model, the Huff NetLogo model is an ABM, an agent-based model. Stores, establishments, in Huff NetLogo model are represented by patches whose `store-at?` properties are set (dynamically) to `true`. Customers are represented by turtles. The number of customers is controlled by the customers slider on the Interface tab, shown in Figure 6.1. The number and placement of stores is, as already described, determined at run time by the user working interactively with the model. In NetLogo, both patches and turtles are agents. As such, they are potentially rich in capabilities and capacities to act and to store information. Turtles may move about in the NetLogo world, although they do not do so in the Huff model. Patches are fixed in place, but otherwise are potentially very active and capable agents. NetLogo's world is based upon a two-dimensional grid of patches, arrayed much as is a checkerboard. Each patch has an address. by default the patch at the center of the world has address 0, 0. Other addresses work like Cartesian co-ordinates with respect to it. For example, in Figure 6.1 reference is made to a store at store location [7 8]. This is the black square (store) that is east of the three other black squares. It is seven patches east of 0, 0 and eight patches north.

6.1.5 Employment 1

The program file for the Huff NetLogo model is *Huff.nlogo.* It is available at the book's web site and at the NetLogo Modeling Commons `http://www.modelingcommons.org`. To run the program you will need to download and install NetLogo on your computer. The NetLogo home page is `https://ccl.northwestern.edu/netlogo/`. There, you can download NetLogo and access a rich corpus of information about NetLogo.

The Huff model, with the customer and store data associated with Figure 6.1, gives us the expected store traffic for each of the four stores present. The NetLogo implementation affords exploration of the basic result. One way to do this is to keep the customers fixed and vary the locations of the stores. For example, imagine that three stores are presently in operation, those at [7 8] (which is your store), [-13 8], and [-5 14]. The present situation is as shown in Figure 6.2. Your store has traffic of about 430, which is expected to drop to about 258 once the new store at [2 -1] comes online. Given this, you might consider relocating your store. Where would be a more advantageous position for your store? Using the add-drop-stores and calculate-store-traffic buttons in the Huff NetLogo model you can explore for possibilities. For example, if you relocate to [-9 -3] your expected traffic

is 276, an improvement over the 258 you expect if you stay. See Figure 6.3.

In light of the Huff model, a second way to improve your store's traffic numbers is to increase its size. Starting with the model configuration shown in Figure 6.3, remove the new store at [-9 -3] by (1) clicking on the add-drop-stores button, (2) clicking on the store's patch (which makes the store disappear, and (3) clicking again on the add-drop-stores button. Now you can re-create your original store as follows:

1. Type `inspect patch 7 8` after `observer>` in the Command Center then press Enter/Return on your keyboard.

   This will produce the window shown in Figure 6.4.

2. Edit `pcolor` ("patch color") to change it from 9.9 (white) to 0 (black; you can actually type `black` but NetLogo will change it to 0).

3. Edit `store-at?` to change it from `false` to `true`.

4. Edit `store-size` to change it from 1 to 2.

5. Click on the calculate-store-traffic button.

Figure 6.5 shows you what you should get in result.

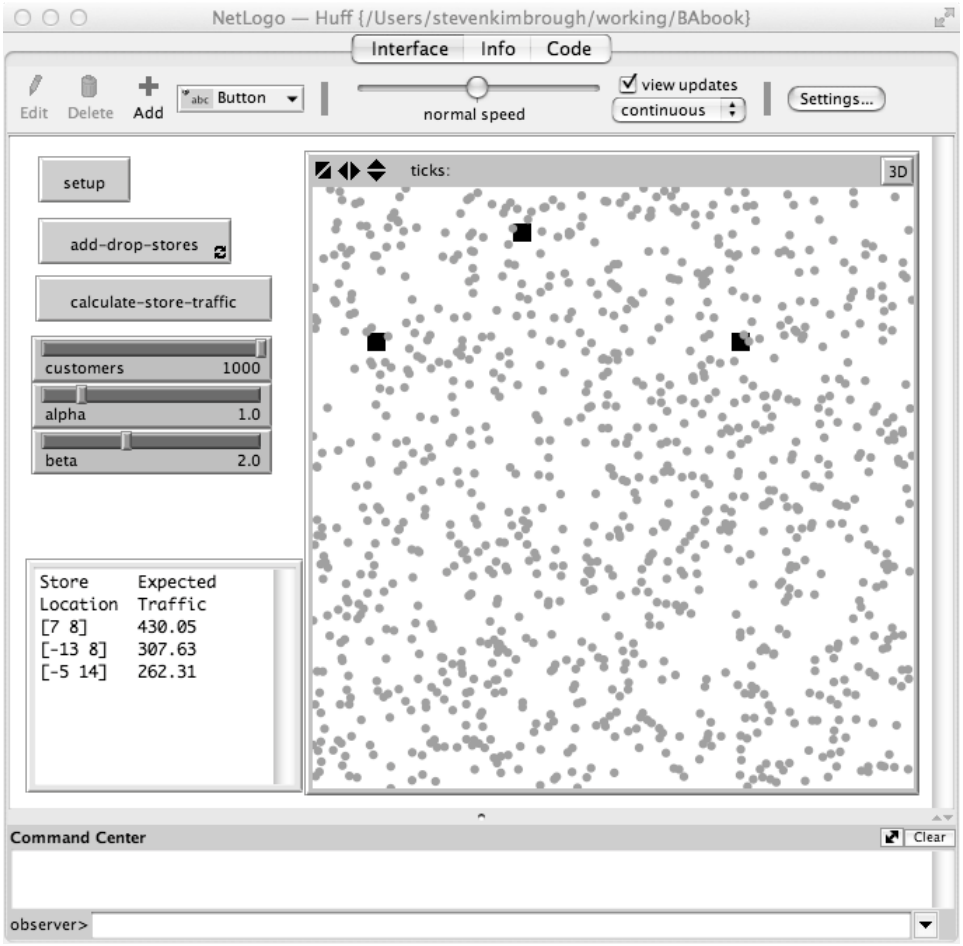

Figure 6.2: Huff, a NetLogo implementation of Huff's model.

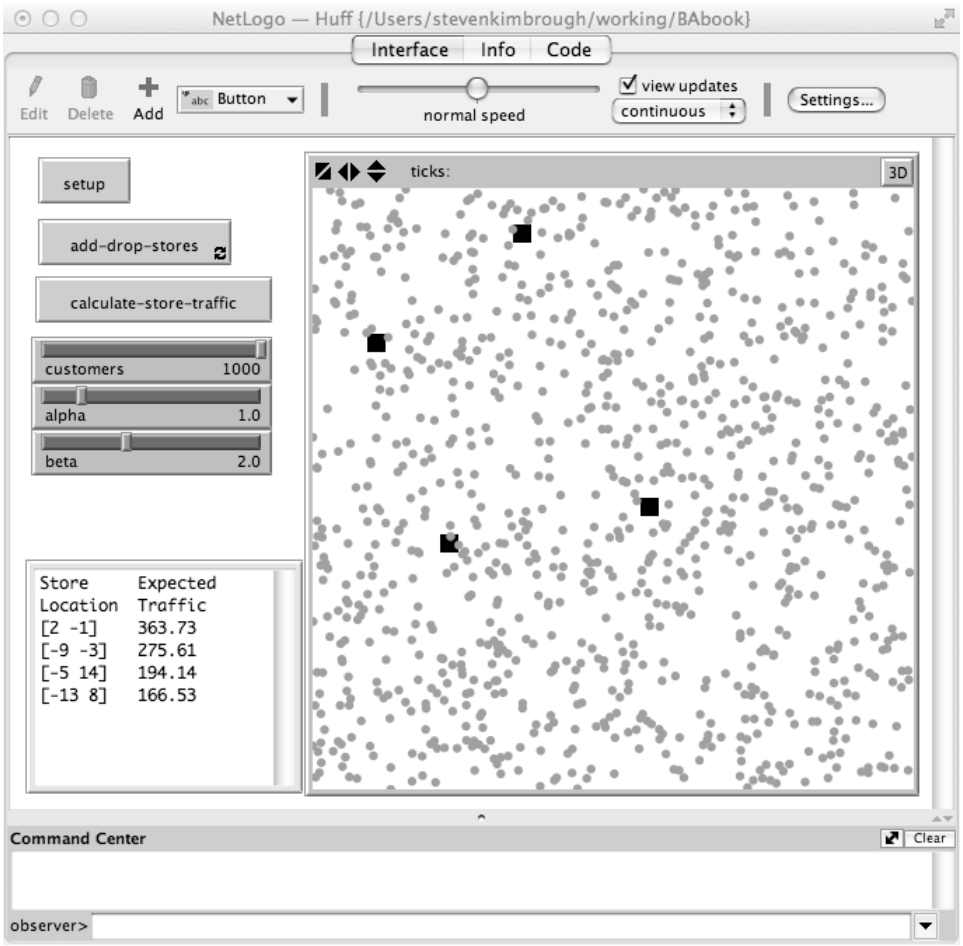

Figure 6.3: Huff, a NetLogo implementation of Huff's model. Results of a possible move from [7 8].

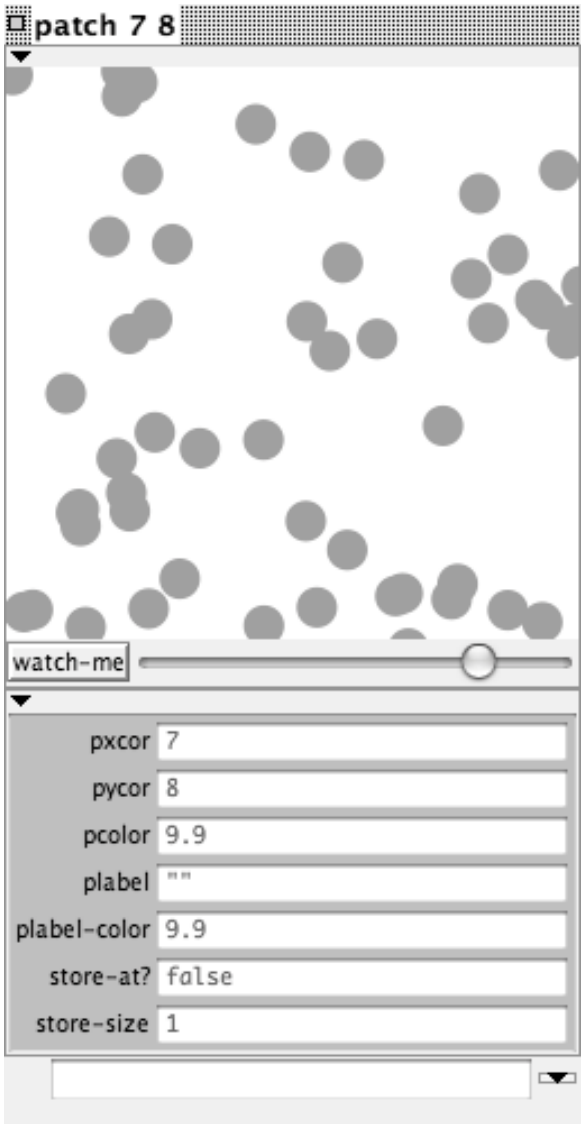

Figure 6.4: Inspecting the patch at position [7 8] of the Huff NetLogo model.

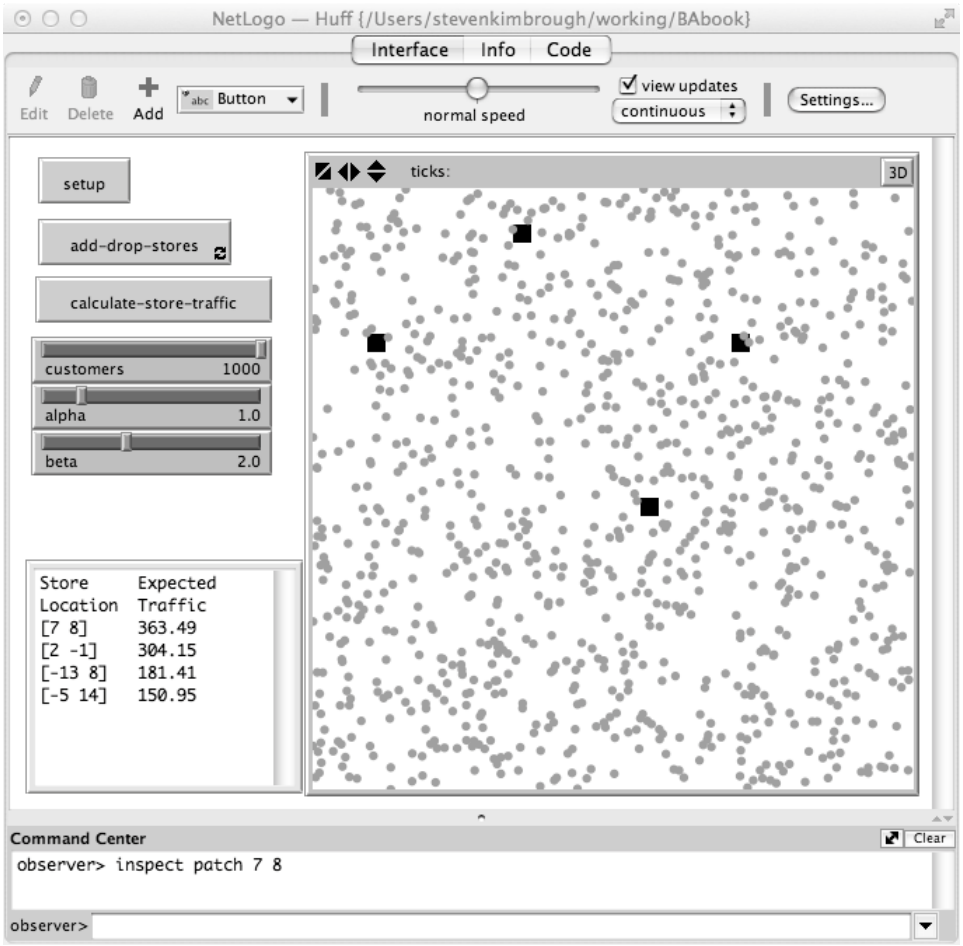

Figure 6.5: Huff NetLogo model with size 2 for the store at 7 8, and otherwise the original values.

6.1.6 Design & Encoding 2

The Huff Turtle-Based NetLogo model, Figure 6.6, reimplements the Huff NetLogo model with one crucial design difference: stores are now represented as NetLogo turtles rather than as NetLogo patches. (Trivially, but usefully, stores are presented as buidling-suggestive icons, rather than as solid squares.) Notice that the basic setup and results exactly duplicate Figure 6.1 for the original Huff NetLogo model. Having stores be turtles makes it much easier to program them to move about, for patch agents in NetLogo cannot move. This presents us with opportunities to add useful features to the NetLogo model. Figure 6.6, in consequence, has the look of a genuine analytics dashboard. We shall explore the new features in the next section.

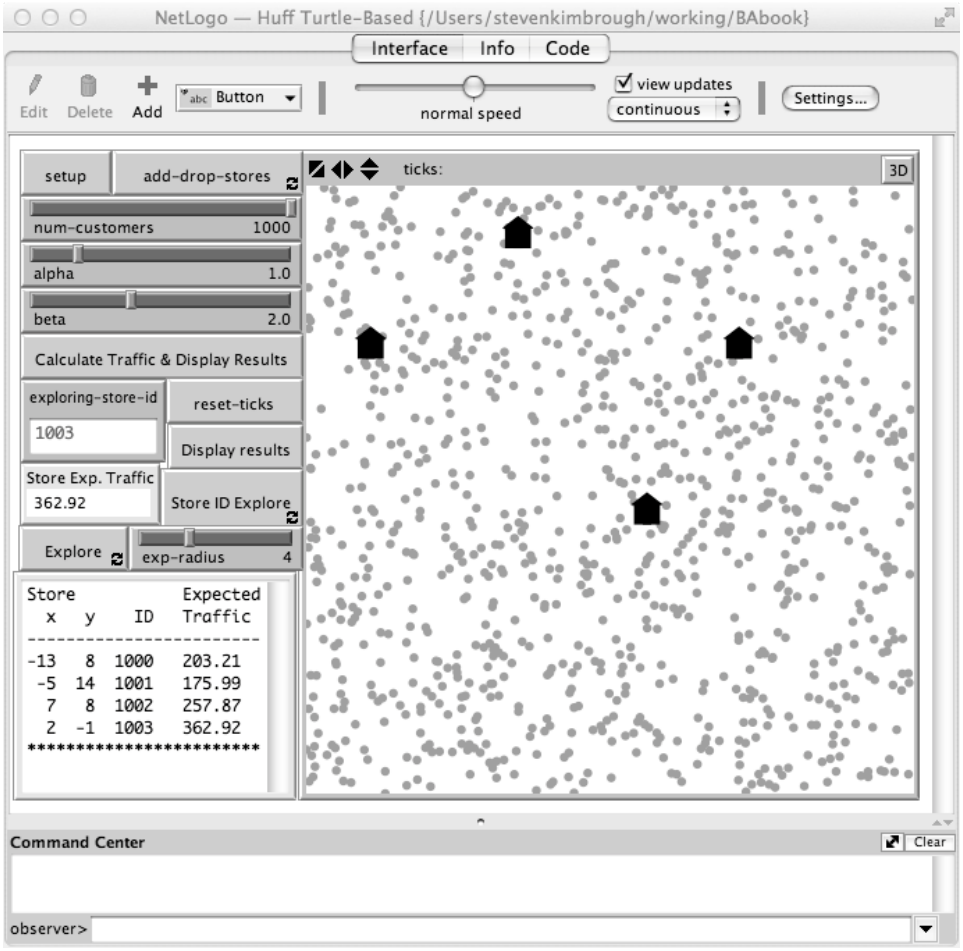

Figure 6.6: Huff Turtle-Based model, a NetLogo implementation of Huff's model. The basic setup and results exactly duplicate those of the Huff model in Figure 6.1

6.1.7 Employment 2

The program file for the Huff Turtle-Based NetLogo model is *Huff Turtle-Based.nlogo.* It is available at the book's web site and at the NetLogo Modeling Commons `http://www.modelingcommons.org`. To run the program you will need to download and install NetLogo on your computer. The NetLogo home page is `https://ccl.northwestern.edu/netlogo/`. There, you can download NetLogo and access a rich corpus of information about NetLogo.

Let us reconsider in our new environment the post-solution analysis and deliberation case we explored for the original Huff NetLogo model (in §6.1.5): three incumbent stores share the current environment, a new store is known to be coming in, what are the consequences and what might be done?

1. Edit the input widget exploring-store-id so that it contains 1002.
2. Click the reset-ticks button.
3. Adjust the exp-radius slider to have a value of 3.
4. Click the Store ID Explore button.
5. Watch while the store moves around and the tick counter increases.
6. Click the Store ID Explore button to halt search when `ticks` reaches, say, 40.
7. Click the Display results button.

When you are done the results should look something like Figure 6.7 (Of course your results will differ because you are using a different stream of random numbers.) Notice that store has improved its traffic position significantly, to 315, much better than the 276 we found in Figure 6.3. How did store 1002 do it? Time proceeds by ticks (or a notional clock). At each tick, the store randomly picked a spot within a radius of three patches (the value of `exp-radius`) from its current position. It then had the global store traffic calculations redone. If the store 1002's expected traffic increased with the move, it retained the move; otherwise, it moved back to where it was at the start of the tick and restored the original traffic values. This completes the tick. The process was repeated until we stopped it at 41 ticks. For future reference, this procedure is an example of *greedy hill climbing*. See Chapter **??** for more information on greedy hill climbing. In the interim we hope it is obvious to you that this search procedure is aptly named.

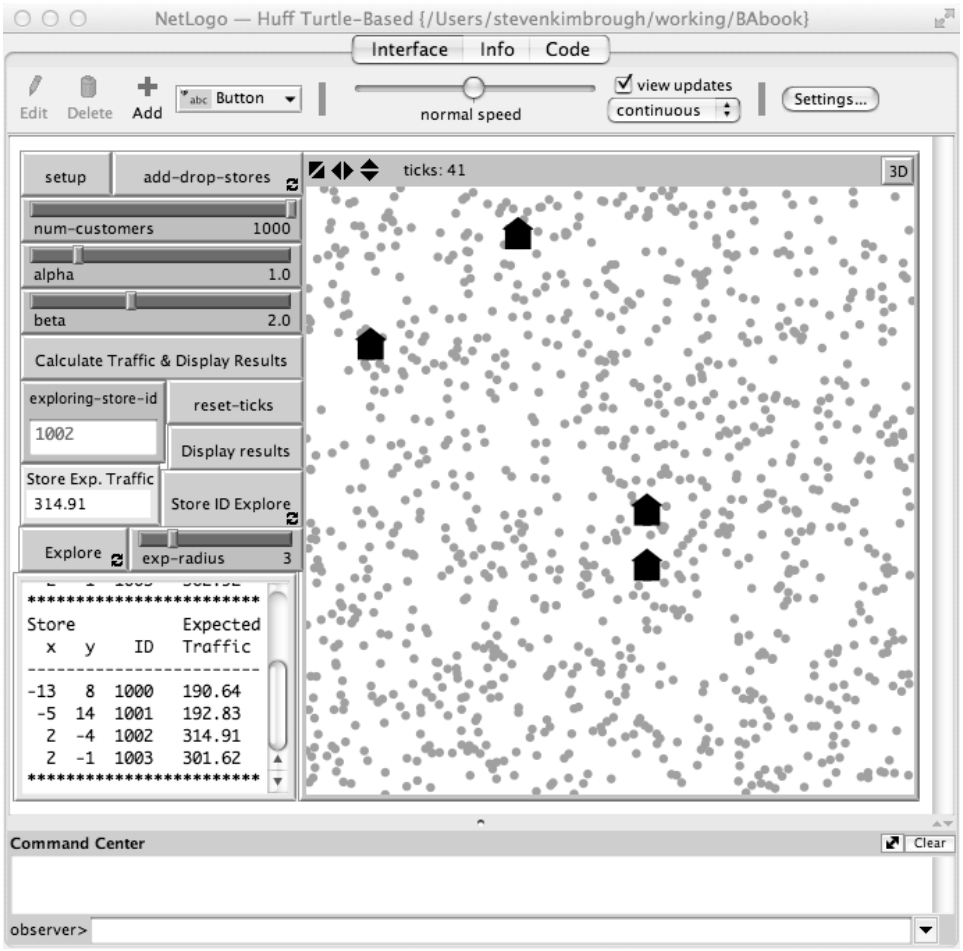

Figure 6.7: Huff Turtle-Based model, configuration descended from that of Figure 6.6 after exploration by store 1002.

Now let's see what happens if all of the stores engage simultaneously in searching for better deals for themselves.

1. Type `ask store 1002 [setxy 7 8]` at the `observer>` prompt and press Enter/Return on your keyboard. This restores 1002 to its original position.
2. Click the Calculate Traffic & Display Results. This restores the output widget to its original message.
3. Click the reset-ticks button. This starts our clock anew.
4. Retain the exp-radius slider at the value of 3.
5. Click the Explore button.
6. Watch the stores move as the ticks counter increases.
7. Click the Explore button again after movement has halted. We did this when `ticks` equaled 27 (resulting in it stopping at 28).
8. Click the Display results button.

Your results should look something like Figure 6.8.

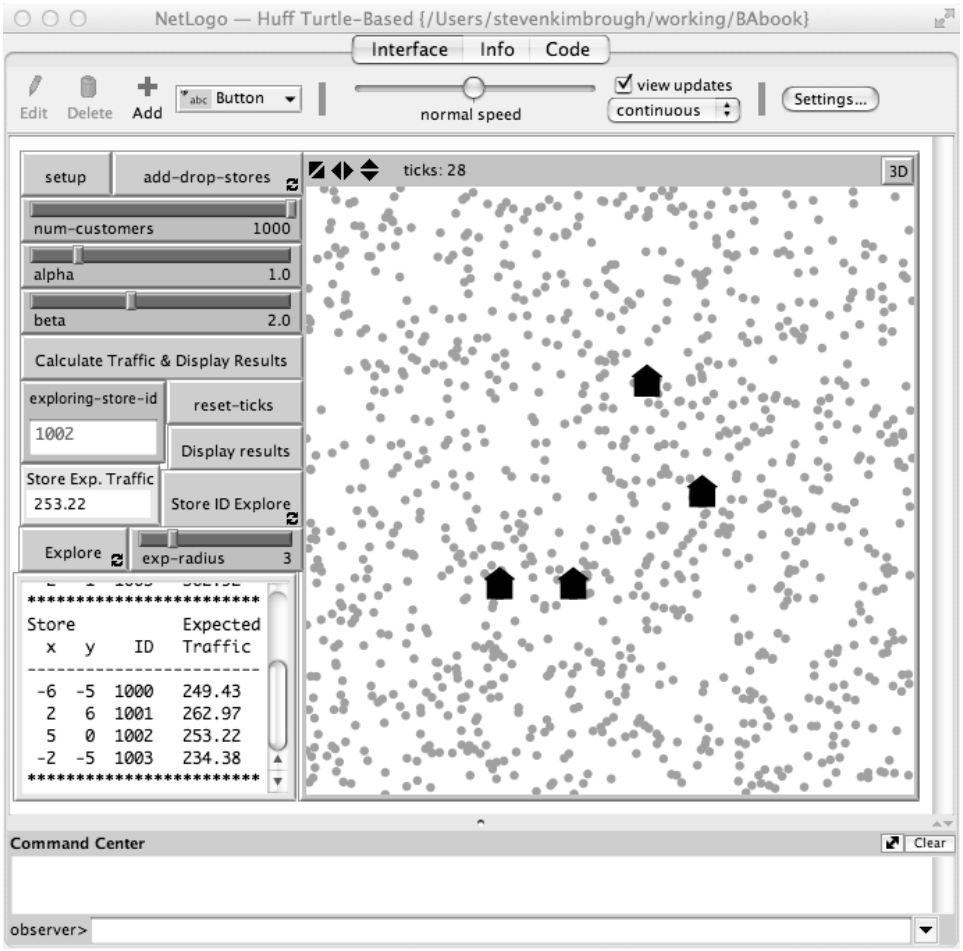

Figure 6.8: Huff Turtle-Based model, configuration descended from that of Figure 6.6 after exploration by all stores simultaneously.

Finally, we repeated the last experiment after doubling the size of store 1001 to 2, keeping the other stores at size 1. Figure 6.9 shows the results.

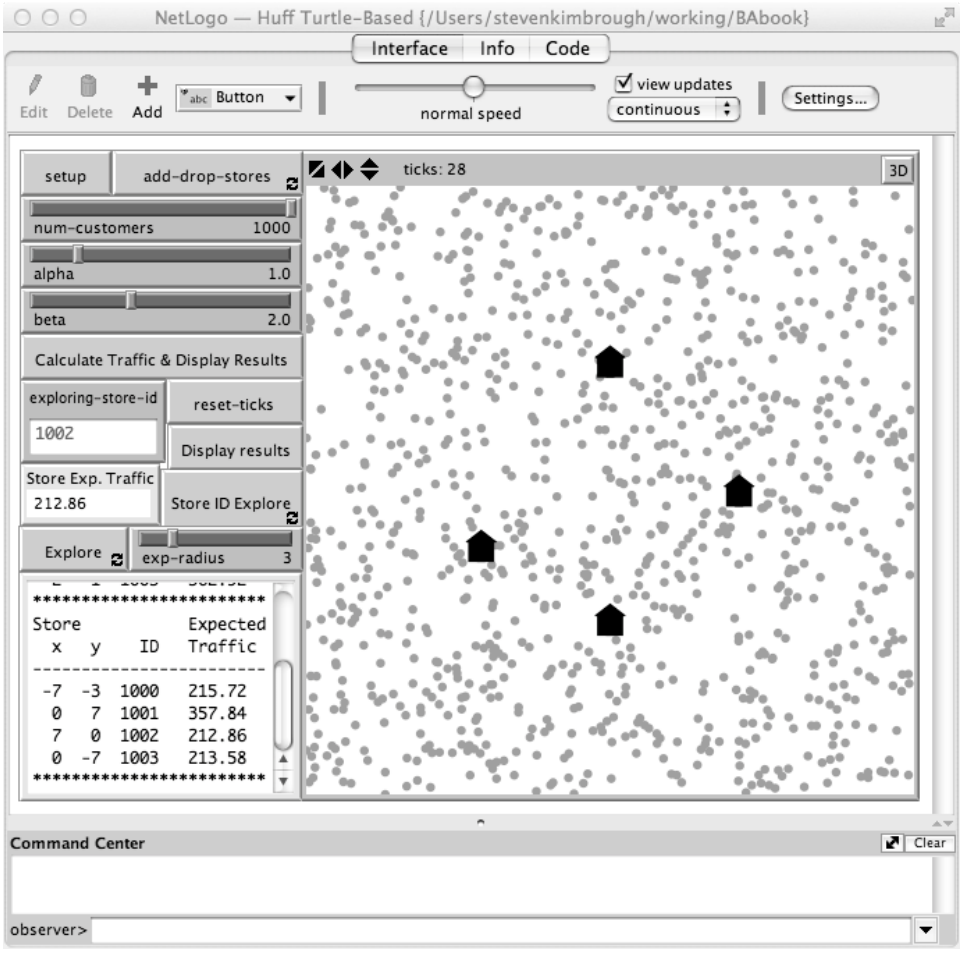

Figure 6.9: Huff Turtle-Based model, configuration descended from that of Figure 6.6 after exploration by all stores simultaneously but with store 1001 having a size of 2 (with 1 for the others).

6.2 Discussion

Decision making contexts can be divided cleanly into two kinds. In *parametric* decision making the decision maker has a number of choices, picks one of them, and receives a reward (positive or not) depending upon how Nature disposes. Nature may be capricious, may behave randomly, but Nature is indifferent to the decision maker. Nature does not have her own interests nor does she care about the interests of anyone else. In *strategic* decision making there are always at least two decision makers. Any one decision maker has a number of choices, picks one of them, and receives a reward (positive or not) depending upon how Nature disposes *and* on how any other decision makers choose. So, strategic decision making is also aptly known as *interdependent decision making* because the decision makers are dependent upon each other, in part, for what happens.

Converse's formula, or at least our use of it, is straightforwardly for parametric decision making. There is city A with its existing store, what will be our traffic if we install a store in city B? The answer for the question as formulated depends only on Nature. With our treatment of the Huff model, especially in the Huff Turtle-Based NetLogo implementation, we are well within the territory of strategic decision making.

With these brief preparatory remarks to hand we continue the discussion in the next section.

6.3 For Exploration

1. Under what sorts of conditions is it likely that an organization facing a store location problem can do its market area analysis largely without regard to strategic considerations (that is, without having to take into account possible decisions by competing firms)?

2. Under what sorts of conditions is it likely that an organization facing a store location problem can do its market area analysis largely without neglecting strategic considerations (that is, without being able to ignore possible decisions by competing firms)?

3. Use the Huff NetLogo model to find the best position you can for the store that is originally located at [7 8]. That is, repeat the experiment we reported in Figure 6.3. What do you find?

4. Consider the generally good positions that you see for the store in the

experiment we reported in Figure 6.3. Characterize them. Do you see any patterns?

5. What justification can you give for the positive role of store size in the Huff model?

6. Characterize and explain the result of the experiment resulting in Figure 6.7.

7. Characterize and explain the result of the experiment resulting in Figure 6.8.

8. Characterize and explain the result of the experiment resulting in Figure 6.9.

9. Characterize and compare the results of the experiments behind Figures 6.8 and 6.9.

10. Read up on the Hotelling model(s) for store location. See in particular the *Hotelling's Law.nlogo* model, which is in the Models Library that ships with NetLogo. Compare and contrast what we learn from it versus the Huff model implemented as *Huff Turtle-Based.nlogo.*

11. Design new experiments to conduct with the Huff Turtle-Based model to explore the effect of store size on the outcome of exploratory search by all stores.

12. Design new experiments to conduct with the Huff Turtle-Based model. Look at more or fewer stores, different values of α and β, non-random distributions of customers, and so on.

13. Discuss how you might go about conducting a real world study using the Huff Turtle-Based model. How would you map real customer locations to the model? What sorts of problems might you expect to have in building such a model? Under what conditions do you think the resulting model, admittedly imprecise in its representation, would be useful?

6.4 For More Information

There, in the *NetLogo/* directory you will find the NetLogo programs we mention in conjunction with the chapter: *Converse Formula.nlogo* and *Huff*

Interactive.nlogo. These files are also available online at `http://www.modelingcommons.org`.

The NetLogo home page is `https://ccl.northwestern.edu/netlogo/`. There, you can download NetLogo and access a rich corpus of information about NetLogo.

On market area analysis, and the Huff gravity model:

1. `http://www.directionsmag.com/articles/-retail-trade-area-analysis-using-the-huff-model/123411`
2. Huff, D.L., 2003. Parameter Estimation in the Huff Model. ArcUser, October-December, 34–36. `http://www.esri.com/news/arcuser/1003/files/huff.pdf`
3. Two videos about using the GIS Maptitude to do Huff models.

   `http://www.youtube.com/watch?v=9d0Ccsj2Ct4`

   `http://www.youtube.com/watch?v=pOfz4d2un28`

Foundations of Location Analysis, edited by Eiselt and Marianov, has an excellent historical chapter pertaining to the Huff model as well as Converse's formula (**?**, Chapter 18).

The literature on agent-based modeling is burgeoning. Still very much worthwhile and highly recommended is *Growing Artificial Societies* by Epstein and Axtell **?**. *Agent-Based and Individual-Based Modeling* by Railsback and Grimm **?** provides excellent tutorial material and is based in NetLogo.

File: Huff.tex

Chapter 7

Making Forecasts: Group Decision Forecast (GDF) Problems

7.1 Introduction

It is often the case that interested parties have identified a group of decision makers for which they wish to **forecast** the **outcome** of a deliberation.[1] Examples include senior management in a firm who must decide whether to launch a new product and if so which one(s), legislators in a committee who must put together a budget bill, national leadership deciding on foreign policy, and so on. Cases such as these present what we shall call *group decision forecast* (GDF) problems. The interested parties will often construct models when faced with GDF problems. This chapter introduces GDF modeling.

[1] In ordinary language, "prediction," "forecast," "extrapolation," and "prognostication" are used more or less synonymously. Those in the modeling trade, however, tend to use these words with distinct meanings. Calling something an extrapolation (based on a given model) is typically to say merely that this is output of the model given recent data, projected into the future. There is no implied assertion that such a projection is valid. If things don't change, this is what we will have. A prediction is an assertion about something unobserved in which the person making the assertion is making something of a "reputation bet." If the prediction is very much off, the person stands to be embarrassed because the person has stood behind it. "Forecast" normally is used, as it is here, for an intermediate form lying between prediction and extrapolation. With more evidence we might turn a forecast into a prediction. Weathermen make forecasts, not predictions; they tell us what their models are saying. "Prognostication" is a more informal term for referring to what the model says will happen.

The target decision making groups for a GDF model are composed of **agent**. The agents in these models have different values, positions, views, and powers, which they bring to a collective decision making process that chooses among available options. forecast of this sort can be, have been, and are being used to support deliberation by those within the modeled groups and those affected by the modeled groups. The scope of application and usefulness of these kinds of models is very broad.

The aim of this chapter is to introduce the use, application, construction, and analysis of GDF models. These models, as we shall see, typically rely upon one or more **generalized voting model** to construct their forecast of what the outcome of the group's deliberations will be. Here we will focus on a set of particular classes—or **use case**—of voting models and their analysis. By doing so, we aim to help the reader acquire an informed intuition for what voting analysis can do and why it is valuable. It must be remembered that there is no magic bullet. What we are presenting here is a systematized way of approaching these problems in order to arrive at an understanding.

7.2 Use Case: Position Bargaining

This section develops and discusses models that can forecast the outcome a group of negotiating agents might choose when the group engages in position bargaining.

7.2.1 Position Bargaining Chacterized

In position bargaining, a number of agent collectively negotiate to decide which of a number of options, ordered on a single spectrum or scale, will be chosen. (We note that position bargaining is referred to in the literature by other names, such as one-dimensional spatial bargaining, etc.; see (**?**, §§5–6).) In modeling position bargaining, each agent begins the bargaining advocating a particular outcome. That advocated outcome (which may or may not be sincerely held) is called the **position** of the agent and it is assigned a numerical value according to its place in the ordered spectrum of all options (the range of advocated outcomes). Each agent has a position, and some agents may share the same position. The possible outcomes of the bargaining are, or at least include, the position of the several agents.

In addition to its position, each agent has a score for **salience** and for **influence**. Salience for an agent with respect to an issue is an indicator

of how important the issue is to the agent. Influence for an agent with respect to an issue is a measure of the degree to which the agent is able to affect the outcome: it is a measure of power relative to the other agents. Finally, each agent has a level of **exercised power** affecting the decision process. exercised power for an agent with respect to an issue is a derived quantity; it is an amalgam of the agent's salience and influence. The idea is that influence measures the potential power of an agent to affect the results of the bargaining and salience measures the agent's willingness to exercise that power. Together, they combine to give us an index of the agent's actual sway, its exercised power, in influencing the negotiations.

We would like to know what the outcome of the bargaining will be. Our position bargaining models initially will use the position and exercised power of the agents to make forecasts. Later, we will avail ourselves of other information. Our practice is to proceed incrementally in the exposition.

In discussing the position bargaining use case, we will base our explanation on an example taken from the academic literature. Bueno de Mesquita **?** provides a well-known instance of using voting models to forecast negotiated outcomes. We are, of course, aware of controversy attached to this paper; resolving this controversy is well beyond the scope of this document. Instead, we present **?** as an example (a particularly accessible example) of how modeling can be, and in fact is, claimed to provide insight on the important area of forecasting negotiation outcomes. Published during the Iran-Iraq War (1980–1988; aka the First Persian Gulf War) and seeking to model decision making on the Iranian side, the paper identifies 27 agent of interest (which he calls groups), of which 18 are actually used in his model to predict the position eventually arrived at by these agents. Each of the agents is given a two or three letter abbreviation for its name.[2] We shall use these abbreviations in what follows to identify the agents in the models; for the purpose of our explanation the true identities of these agents are not important. In his paper Bueno de Mesquita forecasted that the war would continue but in a more economical fashion, with the hardliners of Ayatollah Montezari, the Tehran Militant Clerics and the Qum Clerics sidelined. In the end, the fighting on the battlefield ended only in August 1988 with Resolution 598, a UN-brokered ceasefire, but the hostilities continued into the peace-negotiating chamber until a peace was finally signed

[2]E.g., MON for Ayatollah Montezari, TMC for Tehran Militant Clerics, QUM for Qum Clerics, SC for Supreme Court, LCR for Lower Class Rural Peasants; see (**?**, Table 1) for the full listing.

in 1990.

In line with our description of how agents hold opinions, Bueno de Mesquita **?** assigns three numeric data values to each agent. He calls them *issue position, salience,* and *influence.* These map to our introduced terminology. What we call exercised power will be modeled as the product of the influence and salience values from Bueno de Mesquita's data.

7.2.2 Basic Data

The data from **?** are collected in our Table 7.1 below, and constitute the input for our models to follow.[3] Our discussion will proceed with the data as is; the methods of capturing these data are beyond the scope of this discussion. Note that column A enumerates a rank ordering of the position (column C), lowest to highest. We will also use these numbers as indexes to indicate the associated agents in column B. The issue positions (Bueno de Mesquita's term; our position) and salience values are indicated in **?** Table 2 by tick marks on non-numeric scales. We derived the actual values by direct measurement on the diagrams.

[3]The agent (group) abbreviations and the influence scores for the agents (taken from (**?**, Table 1)) constitute columns B and E of our Table 7.1. Columns C and D of our Table 7.1 contain data taken by measurement from Table 2 of **?**. These are the data reproduced in our Figures 7.2 and 7.1, with column C holding data from Figure 7.1 and column D holding data from Figure 7.2. The SC group does not appear in this scale in the original paper (**?**, Table 2). Instead, SG, which is otherwise absent from the paper, appears at position 11.1. We assume this is simply a typo and have recorded it as SC.

A	B	C	D	E
Order	Agent	Position	Salience	Influence
1	UMC	0.0	0.8	1.1
2	TEC	0.0	2.5	0.6
3	MCR	2.1	2.5	0.9
4	SOV	3.0	10.3	0.6
5	PM	4.7	6.8	9.0
6	BAZ	5.1	0.8	5.6
7	REV	5.1	8.5	12.4
8	JC	5.1	10.3	11.3
9	PRE	6.4	10.3	10.7
10	UP	7.3	2.5	4.5
11	LCR	7.3	3.4	4.5
12	KUR	8.6	4.3	2.3
13	COM	9.8	8.5	11.8
14	SC	11.1	6.0	4.5
15	CG	11.1	4.3	9.0
16	MON	12.4	9.4	0.1
17	TMC	12.4	9.4	3.4
18	QUM	12.4	9.4	4.5

Table 7.1: Data drawn from **?** Tables 1 and 2.

Figure 7.1 reconstructs the position scale from the paper, adding actual numerical values for the scored items. There we see that position of the agents are arrayed on a phrase-anchored line entitled “Issue: What is an acceptable settlement of the war with Iraq for each of the groups?”. In that scale (on the line), the phrase “End War Mediated Settlement” anchors the low end, “Continue War with Goal of Overthrowing Hussein” anchors the high end, and “Continue War Resolve Economically” hovers in the middle.

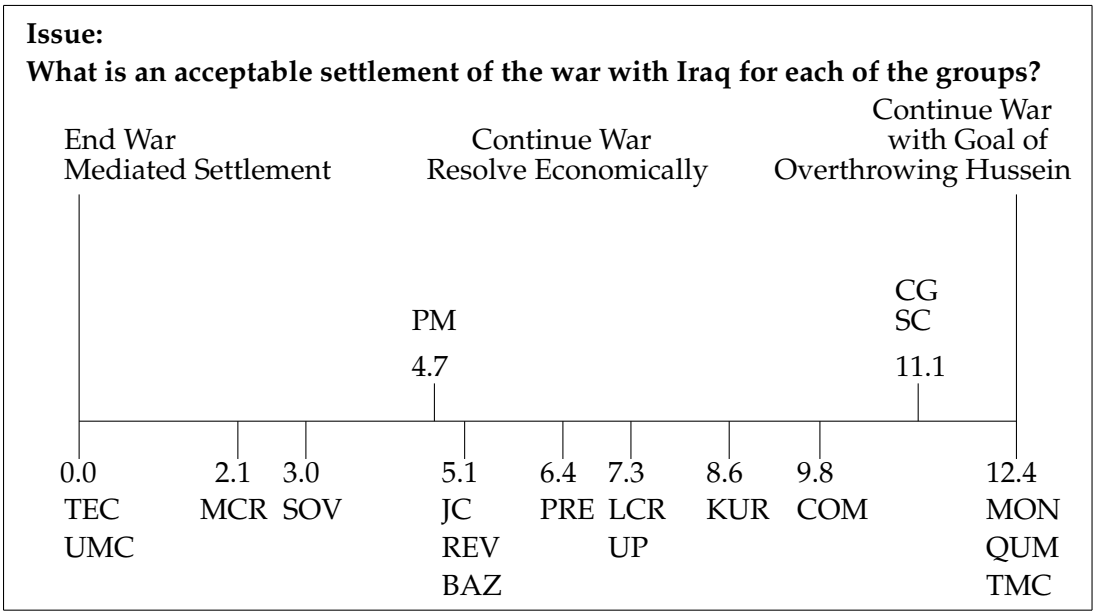

Figure 7.1: Issue position scores (after **?** Table 2).

Figure 7.2 reconstructs the salience scale from the paper, adding actual numerical values for the scored items.

Finally, the influence scores in (**?**, Table 1) were quantified from 0 to 12.4. The exact underlying scale is not given and can be taken as arbitrary for present purposes. What is important is the agent's relative score within the scale. Of the 25 groups with non-zero scores on influence, 18 are used in the paper's subsequent analysis. This is why there are 18 agents identified in Table 7.1, and Figures 7.1 and 7.2.

The paper **?** forecasted a possible outcome as a single position on the line (7.3, the position of both LCR and UP), with the intent of informing discussion. Unfortunately, it did so without providing the underlying voting model. In what follows, we illustrate the present use case, position bargaining, by applying and exploring several different voting models to the data given in **?**.

At a high level the setup is quite simple. First, there are agent having some say in the outcome. These agents may be any combination of individual people or groups or institutions. Second, the possible outcomes are represented as position on a line segment, and are scored numerically. For example, agent MCR holds the position of 2.1 (suggesting a wish for a me-

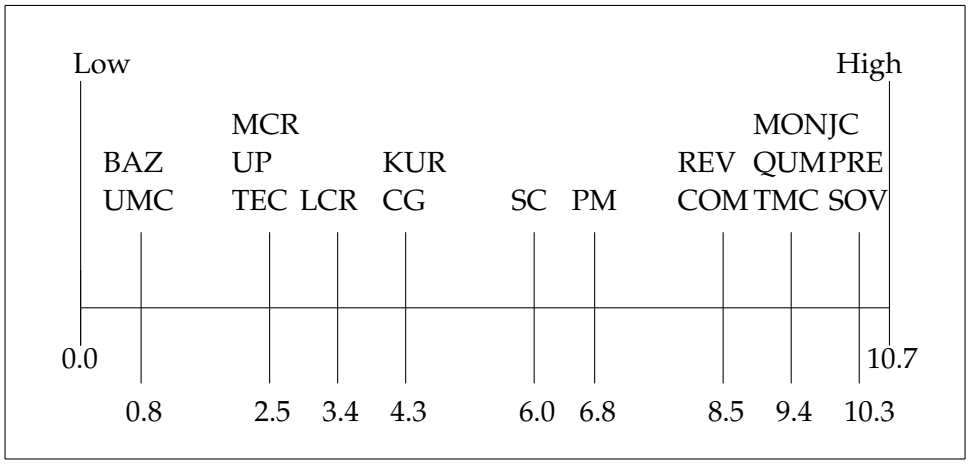

Figure 7.2: Salience scores (after **?** Table 2).

diated settlement), KUR the position of 8.6 (suggesting a more vigorous attitude to the war), and so on. Figure 7.1 displays the position scores of all of the agents in this example. Third, every agent has an exercised power score, which in the **?** example is the product of the agent's influence score and its salience score (Figure 7.2). With the position, salience, influence and exercised power scored for each agent, a **voting model** can be employed to generate a forecast outcome. This outcome is represented numerically as a location (position) on the line segment, just as the initial position were. In short, the models can use these data to forecast which of the possible outcomes the agents will choose, as we shall now demonstrate.

7.2.3 Building and Solving the Models

We will explore here four of the many possible models for forecasting what the outcome of deliberation on war policy by these 18 groups could be.

7.2.3.1 Model 1: Weighted Median Position of exercised power

This model is inspired by the median voter theorem. In this, we array in order on a line the various position held and for each such position we record the number of voters who hold it. The median voter theorem then says that the median position (taking into account the number of agents holding each position) will be the outcome of the voting process.

In our model we think of each position taken as arrayed in order on

a line and having associated with it a certain voting strength. We resolve the negotiation by assuming that the median position in terms of strengths will prevail. In this particular example, we use the exercised power scores of individual agents instead of the number of voters at each position as the indication of voting strength. We find the associated median position, and we forecast it as the outcome.

exercised power here represents the amount of power the agent will[4] bring to bear in the negotiation. We define e_i—the exercised power of agent (group) i in swaying the results of the bargaining—as $S_i \times I_i$, that is as the product of the agent's salience and influence (Columns D and E of Table 7.1). Table 7.2 displays the agent exercised power in column D. Column E presents the accumulated sums of the exercised power values in column D, starting at the top with agent ID 1 (with minor apparent differences due to rounding of the underlying numbers). The accumulated total for all the agents is 686.1, associated with the last agent, 18. The weighted median position is the position associated with half of this value, that is 343.1 (with rounding). Examining column E we see that the interval running from 298.3 through 408.5 contains this value and so the weighted median position is PRE. This is indicated by a check mark in column F. PRE, agent 9, is the model's declared winner.

[4]We emphasize the intended meaning of exercised power in this context. It represents a measure of power actually applied, not merely the potential power that could be applied.

A	B	C	D	E	F
Order	Agent	Position	Exercised Power	Accumulated Sum of e_i	Forecast Rule
ID	$\mathcal{A}$	Θ	e_i	$\sum_{i=1}^{\text{ID}} e_i$	Weighted Median
1	UMC	0.0	0.9	0.9	
2	TEC	0.0	1.5	2.4	
3	MCR	2.1	2.3	4.6	
4	SOV	3.0	6.2	10.8	
5	PM	4.7	61.2	72.0	
6	BAZ	5.1	4.5	76.5	
7	REV	5.1	105.4	181.9	
8	JC	5.1	116.4	298.3	
9	PRE	6.4	110.2	408.5	✔
10	UP	7.3	11.3	419.7	
11	LCR	7.3	15.3	435.0	
12	KUR	8.6	9.9	444.9	
13	COM	9.8	100.3	545.2	
14	SC	11.1	27.0	572.2	
15	CG	11.1	38.7	610.9	
16	MON	12.4	0.9	611.9	
17	TMC	12.4	32.0	643.8	
18	QUM	12.4	42.3	686.1	

Table 7.2: position of the weighted median of exercised power using data from (**?**, Tables 1 and 2). Forecasted outcome = 6.4, the position associated with agent PRE.

It will be helpful for purposes of summary and comparison to indicate the main elements of our model in terms of a small framework. The key elements of our model may be summarized as follows.

i. The set of agents. Symbolized as $\mathcal{A}$.

   In the present model, $\mathcal{A}$ = the agents identified in column B of Tables 7.1 or 7.2.

ii. The consideration set of position or possible outcomes. Symbolized as Θ.

   In the present model, Θ = the position identified in column C of Tables 7.1 or 7.2.

iii. The Exercised Power of agent i to weigh in and sway the outcome. Symbolized as e_i.

In the present model, e_i is the product of i's salience and influence (columns D and E of Table 7.1) = $S_i \times I_i$. The e_i values are shown in column D of Table 7.2.

iv. The accumulated sum of the e_i's.

In the present model this is shown in column E of Table 7.2.

v. The forecasting rule. This is the method by which the model, stated in the terms listed above, is to be "solved" and used to generate a model outcome.

In the present model, the forecasting rule takes the position identified by the weighted median exercised power (e_i). The median position is identified by simply taking the median of the cumulative exercised power score and selecting the position bracketing it. As shown in column F of Table 7.2 that exercised power is the one associated with group PRE, number 9 in column A, having a position equalling 6.4. Recall that **?** forecasts 7.3, the position of both LCR and UP. Evidently a different model is being used; the paper does not specify one. In both cases they broadly fall under the phrase anchor "Continue War Resolve Economically".

7.2.3.2 Model 2: Unweighted attractiveness Voting

The implicit theory of model 1 was that the outcome of the group decision process would be determined by the interplay of the exercised power of the agents. In model 2 the implicit theory is that the outcome is determined by a kind of voting process (called *range voting* in the literature). Model 1 emphasizes power at the expense of collective judgment. Model 2 emphasizes collective judgment at the expense of power, and ignores the exercised power of the agents. It calculates **attractiveness** scores for each agent for each and every one of the position that the agents collectively have. attractiveness scores show how attractive each position is to each agent: it is a measure of distance along the position line. The closer a position is to the agent's preferred position, the higher the attractiveness score, until the actual preferred position has a score of 1. Here, the attractiveness scores are totaled up for the position, across every agent, and the maximum total score is used to forecast the group's judgment. Table 7.3 displays the key data and results.

A	B	C	D	E	F
Order	Agent	Position	0-1 Normalized Position	Group attractiveness Sum	Forecast Rule
ID	$\mathcal{A}$	Θ	θ_i	$\sum_{i=1}^{\text{ID}} A_i(\theta_j)$	Maximum attractiveness Sum
1	UMC	0.0	0.00	8.0	
2	TEC	0.0	0.00	8.0	
3	MCR	2.1	0.17	10.4	
4	SOV	3.0	0.24	11.3	
5	PM	4.7	0.38	12.6	
6	BAZ	5.1	0.41	12.9	
7	REV	5.1	0.41	12.9	
8	JC	5.1	0.41	12.9	
9	PRE	6.4	0.52	13.1	✔
10	UP	7.3	0.59	13.1	✔
11	LCR	7.3	0.59	13.1	✔
12	KUR	8.6	0.70	12.7	
13	COM	9.8	0.79	12.1	
14	SC	11.1	0.90	11.3	
15	CG	11.1	0.90	11.3	
16	MON	12.4	1.00	10.0	
17	TMC	12.4	1.00	10.0	
18	QUM	12.4	1.00	10.0	

Table 7.3: Group attractiveness sums for all position; data from (**?**, Tables 1 and 2). Forecasted outcome = 6.4 or 7.3, the position associated with agents PRE, UP, and LCR.

It will be helpful for purposes of summary and comparison to indicate the main elements of our model in terms of a small framework. The key elements of our model may be summarized as follows.

i. The set of agents. Symbolized as $\mathcal{A}$.

   In the present model, $\mathcal{A}$ = the agents identified in column B of Table 7.3.

ii. The consideration set of possible outcomes. Symbolized as Θ.

   In the present model, Θ = the position identified in column C of Table 7.3.

iii. The normalization of the position values to a 0–1 scale. Symbolized as θ_i for agent i.

In the present model, column D holds the 0–1 normalization of the position values from column C. Doing this is simply convenient for the present model; the 0–1 scale is more easily interpreted by the reader. The 0–1 normalization formula is:

$$\theta_i = \frac{\Theta_i - \min \Theta}{\max \Theta - \min \Theta} = \frac{\Theta_i - 0}{12.4 - 0.0} \tag{7.1}$$

iv. The sum across all agents (i's) of their attractiveness scores for the position j in the corresponding row. Symbolized as $\sum_i A_i(\theta_j)$ for agent i and position j.

The Attractiveness (informally, the utility) of possible outcome θ (belonging to the set Θ) for agent i is symbolized as $A_i(\theta)$. It is agent i's score for possible outcome θ. An agent's attractiveness score for a possible outcome is determined by the linear distance between the outcome and the agent's position. Since the agent's position is its (advocated) ideal point, every other distinct position has a lower attractiveness score, depending on its linear distance away. The further a possible outcome is from the agent's position the lower its attractiveness score is for the agent. Formally, in this model we define $A_i(\theta)$ as

$$A_i(\theta) = 1 - |\theta^{*i} - \theta| \tag{7.2}$$

where θ^{*i} is i's (ideal or) position score and θ is the score of any position (both normalized as in column D of Table 7.3).

More specifically, column E values were obtained as follows. The attractiveness to agent i of position θ, $A_i(\theta)$, is $1 - |\theta^{*i} - \theta|$ where θ^{*i} is i's position score (normalized as in column D) and θ is any normalized position. With this definition, we construct a table, T^{APA} (the agent position attractiveness table), in which element $t_{i,j}$ is the attractiveness score of agent i for position j. By definition $A_i(\theta^{*i}) = 1$ and $0 \leq A_i(\theta) \leq 1$ for every θ (assumed to be normalized on a 0–1 range).

The values in column E of Table 7.3 are the *column* sums of the constructed T^{APA}. In our special case, the numbers and identifiers of agents and position are identical, giving Table 7.3 a particularly simple form. In general the number of agents may be different than the number of position or options voted upon.

With m agents and n possible outcomes the constructed table T^{APA} contains $m \times n$ elements, in our case 18×18 = 324 entries, which is why we do not display it in full in this document.

Figure 7.3 shows a portion of the T^{APA} table for the present example. There note, for example, that the PM–MCR entry is 0.79 (emboldened in the table for easy reference). This is the attractiveness value of the MCR position from the perspective of agent PM, whose advocated ideal position is 0.38 (shown in the agent position column). Specifically, $1 - |0.38 - 0.17| = 0.79$. The remaining elements in the T^{APA} table are calculated in the same manner. The column totals in Figure 7.3 correspond to the values of column E of Table 7.3, which are also known as the (unweighted) **range score** in the academic literature.

Agent	Agent Position	All Positions					
		UMC	TEC	MCR	SOV	...	QUM
UMC	0.00	1.00	1.00	0.83	0.76	...	0.00
TEC	0.00	1.00	1.00	0.83	0.76	...	0.00
MCR	0.17	0.83	0.83	1.00	0.93	...	0.17
SOV	0.24	0.76	0.76	0.93	1.00	...	0.24
PM	0.38	0.62	0.62	**0.79**	0.86	...	0.38
BAZ	0.41	0.59	0.59	0.76	0.83	...	0.41
REV	0.41	0.59	0.59	0.76	0.83	...	0.41
JC	0.41	0.59	0.59	0.76	0.83	...	0.41
PRE	0.52	0.48	0.48	0.65	0.73	...	0.52
UP	0.59	0.41	0.41	0.58	0.65	...	0.59
LCR	0.59	0.41	0.41	0.58	0.65	...	0.59
KUR	0.69	0.31	0.31	0.48	0.55	...	0.69
COM	0.79	0.21	0.21	0.38	0.45	...	0.79
SC	0.90	0.10	0.10	0.27	0.35	...	0.90
CG	0.90	0.10	0.10	0.27	0.35	...	0.90
MON	1.00	0.00	0.00	0.17	0.24	...	1.00
TMC	1.00	0.00	0.00	0.17	0.24	...	1.00
QUM	1.00	0.00	0.00	0.17	0.24	...	1.00
Column	Totals:	8.0	8.0	10.4	11.3	...	10.0

Figure 7.3: Portion of T^{APA} table for model 2 (portion within box). Rows correspond to agents; columns under "All Positions" correspond to the positions under consideration. In this model each agent has a single position and the consideration set of positions is just the set of all agent positions.

v. Forecasting rule. This is the method by which the model, stated in the terms listed above, is to be "solved" and used to generate a forecast.

In the present model, the forecasting rule takes the position(s) identified as highest scoring by unweighted range voting.

A little notation will perhaps make this clearer; it will also be helpful in the sequel.

In Figure 7.3 we saw how to construct the T^{APA} table. Column E in Table 7.3 is simply the row sums of the T^{APA} table. We can symbolize this as follows.

$$\text{columnsums}\left(T^{\text{APA}}\right)' = \begin{pmatrix} 8.0 \\ 8.0 \\ 10.4 \\ 11.3 \\ 12.6 \\ 12.9 \\ 12.9 \\ 12.9 \\ 13.1 \\ 13.1 \\ 13.1 \\ 12.7 \\ 12.1 \\ 11.3 \\ 11.3 \\ 10.0 \\ 10.0 \\ 10.0 \end{pmatrix} \tag{7.3}$$

Our forecasting rule then is to forecast the positions associated with the maximal elements of

$$\text{columnsums}\left(T^{\text{APA}}\right)' \tag{7.4}$$

where $'$ indicates the transpose of the associated vector array.

The maximum score in column E is 13.1. It is shared by two different position, PRE at 6.4 and UP–LCR at 7.3. As noted earlier, **?** forecasts the UP–LCR position to be realized.

7.2.3.3 Model 3: Power Weighted attractiveness Voting

Model 3, power weighted attractiveness voting, is like its unweighted cousin, model 2 (unweighted attractiveness voting, in §7.2.3.2), but with the agent-specific attractiveness scores multiplied by the agent-specific weights before the overall sums are taken. Model 2, recall, has an implicit theory that the outcome of the group deliberation can be predicted as a result of a voting process (called range voting) based on the attractiveness scores for each position for each agent. As such and as noted above, model 2 emphasizes collective judgment at the expense of power, and ignores the exercised power of the agent. Model 3 extends model 2 to include consideration of exercised power as well as collective judgment. It does this by calculating for each position the sum, across all agents, of the products of each agent's exercised power score and its attractiveness score. The group attractiveness sums in model 3 are the group attractiveness sums of model 2, weighted by the exercised power scores for each agent.

Table 7.4 displays the key data and results. Compare it to Table 7.3. Column F of Table 7.4 contains the weighted attractiveness scores for the several position.

A	B	C	D	E	F	G
Order	Agent	position	0-1 Normalized position	exercised power	Group Attractiveness Sum	Forecast F
ID	$\mathcal{A}$	Θ	θ_i	e_i	$\sum_{i=1}^{\text{ID}}(e_i \cdot A_i(\theta_j))$	Maximum W attractivenes
1	UMC	0.0	0.00	0.88	275.3	
2	TEC	0.0	0.00	1.50	275.3	
3	MCR	2.1	0.17	2.25	390.7	
4	SOV	3.0	0.24	6.18	439.9	
5	PM	4.7	0.38	61.2	531.0	
6	BAZ	5.1	0.41	4.48	548.5	
7	REV	5.1	0.41	105.40	548.5	
8	JC	5.1	0.41	116.39	548.5	
9	PRE	6.4	0.52	110.21	557.8	✔
10	UP	7.3	0.59	11.25	548.3	
11	LCR	7.3	0.59	15.30	548.3	
12	KUR	8.6	0.70	9.89	529.1	
13	COM	9.8	0.79	100.30	509.3	
14	SC	11.1	0.90	27.00	467.0	
15	CG	11.1	0.90	38.70	467.0	
16	MON	12.4	1.00	0.94	410.8	
17	TMC	12.4	1.00	31.96	410.8	
18	QUM	12.4	1.00	42.30	410.8	

Table 7.4: Group attractiveness sums for all position weighted by agent exercised power; data from (**?**, Tables 1 and 2). Forecasted outcome = 6.4, the position associated with agent PRE.

It will be helpful for purposes of summary and comparison to indicate the main elements of our model in terms of a small framework. The key elements of our model may be summarized as follows.

i. The set of agents. Symbolized as $\mathcal{A}$.

   In the present model, $\mathcal{A}$ = the agents identified in column B of Table 7.4.

ii. The consideration set of possible outcomes. Symbolized as Θ.

   In the present model, Θ = the position identified in column C of Table 7.4.

iii. The normalization of the position values to a 0–1 scale. Symbolized as θ_i for agent i.

In the present model, column D holds the 0–1 normalization of the position values from column C. Doing this is simply convenient for the present model; the 0–1 scale is more easily interpreted by the reader. The 0–1 normalization formula is:

$$\theta_i = \frac{\Theta_i - \min \Theta}{\max \Theta - \min \Theta} = \frac{\Theta_i - 0}{12.4 - 0.0} \tag{7.5}$$

iv. The Exercised Power of agent i to weigh in and sway the outcome. Symbolized as e_i.

In the present model, e_i is the product of i's salience and influence (columns D and E of Table 7.1) = $S_i \times I_i$. The e_i values are shown in column E of Table 7.4.

v. The sum across all agents (i's) of their attractiveness scores for the position j in the corresponding row, weighted (multiplied) by their e_i scores. Symbolized $\sum_i (e_i \cdot A_i(\theta_j))$ for agent i and position j.

The Attractiveness (informally, the utility) of possible outcome θ (belonging to the set Θ) for agent i is symbolized as $A_i(\theta)$. It is agent i's score for possible outcome θ. An agent's attractiveness score for a possible outcome is determined by the linear distance between the outcome and the agent's position. Since the agent's position is its (advocated) ideal point, every other distinct position has a lower attractiveness score, depending on its linear distance away. The further a possible outcome is from the agent's position the lower its attractiveness score is for the agent. Formally, in this model we define $A_i(\theta)$ as

$$A_i(\theta) = 1 - |\theta^{*i} - \theta| \tag{7.6}$$

where θ^{*i} is i's position score and θ is the score of any position (both normalized as in column D of Table 7.4).

More specifically, column F values were obtained as follows. The weighted attractiveness to agent i of position θ, $e_i \cdot A_i(\theta)$, is $e_i \cdot (1 - |\theta^{*i} - \theta|)$ where e_i is the agent's exercised power (column E), θ^{*i} is i's position score (normalized as in column D), and θ is any normalized position. With this definition, we construct a table, T^{AWPA} (the agent weighted position attractiveness table), in which element $t_{i,j}$ is

the weighted attractiveness score of agent i for position j. By definition $A_i(\theta^{*i}) = 1$ and $0 \leq A_i(\theta) \leq 1$ for every θ (assumed to be normalized on a 0–1 range).

The values in column F of Table 7.4 are the *column* sums of the constructed T^{AWPA}. In our special case, the numbers and identifiers of agents and position are identical, giving Table 7.4 a particularly simple form. In general the number of agents may be different than the number of position or options voted upon.

With m agents and n possible outcomes the constructed table T^{AWPA} contains $m \times n$ elements, in our case 18×18 = 324 entries, which is why we do not display it in full in this document. The explanation in Figure 7.3, showing a portion of the T^{APA} table for model 2, applies straightforwardly in the present case.

Specifically, we can construct the 18× 18 E table as follows.

$$E = \begin{pmatrix} e_1 & e_1 & \dots & e_1 \\ e_2 & e_2 & \dots & e_2 \\ \vdots & \vdots & \dots & \vdots \\ e_{18} & e_{18} & \dots & e_{18} \end{pmatrix} \tag{7.7}$$

Then we create the T^{APA} table as before, and

$$T^{\text{AWPA}} = E \otimes T^{\text{APA}} \tag{7.8}$$

where $\otimes$ represents *tabular multiplication* (aka: element-wise multiplication)..

In tabular multiplication the corresponding elements in two tables having identical dimensions are multiplied to produce a third table with the same dimensions. For example:

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \otimes \begin{pmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{pmatrix} = \begin{pmatrix} 7 & 16 \\ 27 & 40 \\ 55 & 72 \end{pmatrix} \tag{7.9}$$

vi. Forecasting rule. This is the method by which the model, stated in the terms listed above, is to be "solved" and used to generate a forecast.

In the present model, the forecasting rule is to forecast the position(s) identified as highest scoring by weighted range voting (with the $A_i(\theta)$ values multiplied by the e_i values and summed for each θ over all i's).

Our forecasting rule then is to forecast the positions associated with the maximal elements of

$$\text{columnsums}\left(T^{\text{AWPA}}\right)' \tag{7.10}$$

where $'$ indicates the transpose of the associated vector array.

We see that it is uniquely maximal at position 9, PRE. Note that we again have a unique forecast, PRE, which happens to be identical to the forecast of our first model, the weighted median position of exercised power from §7.2.3.1.

7.2.3.4 Model 4: Risk Attitude Weighted attractiveness Voting

Model 4, risk attitude weighted attractiveness voting (**?**, page 12), is like model 3 in §7.2.3.3, but it incorporates risk attitude when calculating agent-specific attractiveness scores, the $A(\theta_i)$s. As before these were multiplied by the agent-specific exercised power scores before the overall sums are taken. Table 7.5, column F, contains the weighted attractiveness scores for the several position.

It will be helpful for purposes of summary and comparison to indicate the main elements of our model in terms of a small framework. The key elements of our model may be summarized as follows.

i. The set of agents. Symbolized as $\mathcal{A}$.

   In the present model, $\mathcal{A}$ = the agents identified in column B of Table 7.5.

ii. The consideration set of possible outcomes. Symbolized as Θ.

   In the present model, Θ = the position identified in column C of Table 7.5.

iii. The normalization of the position values to a 0–1 scale. Symbolized as θ_i for agent i.

   In the present model, column D holds the 0–1 normalization of the position values from column C. Doing this is simply convenient for the present model; the 0–1 scale is more easily interpreted by the reader. The 0–1 normalization formula is:

$$\theta_i = \frac{\Theta_i - \min\Theta}{\max\Theta - \min\Theta} = \frac{\Theta_i - 0}{12.4 - 0.0} \tag{7.11}$$

A	B	C	D	E	F	G
Order	Agent	Position	0-1 Normalized Position	Exercised Power	Group attractiveness Sum	Forecast Rule
ID	$\mathcal{A}$	Θ	θ_i	e_i	$\sum_{i=1}^{\text{ID}}(e_i \cdot A_i(\theta_j))$	Maximum Weight attractiveness Su
1	UMC	0.0	0.00	0.88	397.1	
2	TEC	0.0	0.00	1.50	397.1	
3	MCR	2.1	0.17	2.25	510.7	
4	SOV	3.0	0.24	6.18	548.7	
5	PM	4.7	0.38	61.2	606.3	
6	BAZ	5.1	0.41	4.48	616.4	
7	REV	5.1	0.41	105.40	616.4	
8	JC	5.1	0.41	116.39	616.4	
9	PRE	6.4	0.52	110.21	629.5	✔
10	UP	7.3	0.59	11.25	629.2	
11	LCR	7.3	0.59	15.30	629.2	
12	KUR	8.6	0.69	9.89	621.0	
13	COM	9.8	0.79	100.30	606.0	
14	SC	11.1	0.90	27.00	575.1	
15	CG	11.1	0.90	38.70	575.1	
16	MON	12.4	1.00	0.94	530.3	
17	TMC	12.4	1.00	31.96	530.3	
18	QUM	12.4	1.00	42.30	530.3	

Table 7.5: Alternative group attractiveness sums for all position weighted by agent exercised power; data from (**?**, Tables 1 and 2). Forecasted outcome = 6.4, the position associated with agent PRE.

iv. The Exercised Power of agent i to weigh in and sway the outcome. Symbolized as e_i.

In the present model, e_i is the product of i's salience and influence (columns D and E of Table 7.1) = $S_i \times I_i$. The e_i values are shown in column E of Table 7.5.

v. The sum across all agents (i's) of their attractiveness scores for the position j in the corresponding row, weighted (multiplied) by their e_i scores. Symbolized $\sum_i(e_i \cdot A_i(\theta_j))$ for agent i and position j.

The Attractiveness (informally, the utility) of possible outcome θ (belonging to the set Θ) for agent i is symbolized as $A_i(\theta)$. It is agent

i's score for possible outcome θ. An agent's attractiveness score for a possible outcome is determined by a function of the linear distance between the outcome and the agent's position. Since the agent's position is its (advocated) ideal point, every other distinct position has a lower attractiveness score, depending on its linear distance away. The further a possible outcome is from the agent's position the lower its attractiveness score is for the agent. Formally, in this model we define $A_i(\theta)$ as

$$A_i(\theta) = \frac{1 - e^{-R_i(1-|\theta^{*i}-\theta|)}}{1 - e^{-R_i}} \tag{7.12}$$

where θ^{*i} is i's position score, e is the base of the natural logarithm (not to be confused with e_i, the exercised power of agent i), and θ is the score of any position (both normalized as in column D of Table 7.5). R_i is a new parameter. It represents the **risk attitude** of agent i. **?** does not discuss risk, therefore we have assumed risk neutrality and have set R_i to 1 for each agent to obtain the results in Table 7.5. R_i values above 1 are said to indicate risk aversion, values below 1 risk seeking.

More specifically, column F values were obtained as follows. The weighted attractiveness to agent i of position θ is $e_i \cdot A_i(\theta)$, where e_i is the agent's exercised power (column E) and $A_i(\theta)$ is calculated as in expression (7.12) above. With this definition, we construct a table, T^{AWPA} (the agent weighted position attractiveness table) as in model 3, in which element $t_{i,j}$ is the weighted attractiveness score of agent i for position j. By definition $A_i(\theta^{*i}) = 1$ and $0 \leq A_i(\theta) \leq 1$ for every θ (assumed to be normalized on a 0–1 range).

The values in column F of Table 7.5 are the *column* sums of the constructed T^{AWPA}. In our special case, the numbers and identifiers of agents and position are identical, giving Table 7.5 a particularly simple form. In general the number of agents may be different than the number of position or options voted upon.

With m agents and n possible outcomes the constructed table T^{AWPA} contains $m \times n$ elements, in our case 18×18 = 324 entries, which is why we do not display it in full in this document. The explanation in Figure 7.3, showing a portion of the T^{APA} table for model 2, applies straightforwardly in the present case.

Specifically, we create the T^{APA} table as before but using (7.12) to calculate the $A_i(\theta)$ values, and

$$T^{\text{AWPA}} = E \otimes T^{\text{APA}} \tag{7.13}$$

where $\otimes$ represents *tabular multiplication* (aka: element-wise multiplication)..

vi. Forecasting rule. This is the method by which the model, stated in the terms listed above, is to be "solved" and used to generate a forecast.

In the present model, the forecasting rule is to forecast the position(s) identified as highest scoring by weighted range voting (with the $A_i(\theta)$ values multiplied by the e_i values and summed for each θ over all i's).

Our forecasting rule is as in model 3. We forecast the positions associated with the maximal elements of

$$\text{columnsums}\left(T^{\text{AWPA}}\right)' \tag{7.14}$$

where $'$ indicates the transpose of the associated vector array.

We see that it is just barely uniquely maximal at position 9, PRE, with a value of 629.5.

7.2.4 Post-Solution Analysis: General Discussion

It is a commonplace among modelers that the real work for decision making begins *after* a model has been conceived, designed, implemented, tested, validated, and exercised to obtain a solution. It is this *post-solution analysis* phase of modeling that is now the focus of our attention.

Whichever model or combinations of models we decide to use, we shall be interested in obtaining and reflecting upon, as part of our post-solution analyses, a plurality or multiplicity of solutions to our model in the course of doing post-solution analysis. We call this the principle of *solution pluralism*. It enjoins us to identify solutions of interest (SoIs) for purposes of undertaking post-solution analyses. Our solutions of interest in the present case are of several kinds, as will emerge in our discussion.

As a first form of post-solution analysis, we shall be interested in outcomes produced by assumptions that are in an appropriate sense near to those we actually used to obtain the original forecasts. This is the traditional provence of *sensitivity analysis,* which examines the consequences of "small" changes to parameter values in its prototypical case. Further in

this regard, the variance-based sensitivity methods described in **????** and elsewhere, are appropriate, and we think very useful, for investigating the agent position, Θ, and their exercised power, the e_i's.

- *In §7.2.5 we discuss a demonstration implementation of models 1 and 2, called the IranBargaining NetLogo model. That model (software implementation) provides three general post-evaluation services, which we discuss in §7.2.5. Each of the three post-solution analysis services in the IranBargaining NetLogo model (what-if, individual exploration, and group sample exploration) is instrumental for sensitivity analysis understood in this way.*

In a second form of post-solution analysis analysts search for conditions under which certain outcomes or types of outcomes may occur. We call this *outcome reach* (or "What would it take?") analysis.

- *In our present example, PRE is the unique outcome forecast by models 1, 3, and 4, and it is tied with two others in model 2. A question that naturally arises is What would it take to unseat option PRE in favor of another option, say JC? Note that the question does not assume we are looking, as in sensitivity analysis, at only small changes to the model. Sensitivity analysis is about how small changes affect a model's behavior. Outcome reach analysis asks a different question.*

- *We illustrate how to address this question in §7.2.5 in the context of the IranBargaining NetLogo model. There we show that increasing the influence of JC in model 1 will eventually tip the outcome to JC over PRE.*

- *The relevance of what-if services to answering outcome reach questions is clear from the discussion in §7.2.5. A limitation of what-if services for this purpose is that they may require a large number of manually-directed queries to find good answers.*

- *Both individual exploration and group sampling exploration services can, by varying the exploration range percent, yield insights and answers for outcome reach questions. These services have the further advantage, over what-if services, of automatically producing a multiplicity of solutions, affording thereby higher levels of analysis.*

Robustness analysis is a third form of post-solution analysis. In robustness analysis we *seek*, we search for, robust solutions. Given a particular solution to the model, the search for robust solutions may well lead us to new solutions. This is in distinction to sensitivity analysis in which, given

a solution, we seek to assess its robustness by altering parameter values and observing the consequences for the performance of the given solution. Of course, in sensitivity analysis and indeed generally, a solution whose performance does not seriously degrade as parameter values are altered is said to be robust. In short, sensitivity analysis assesses the robustness of a particular solution, robustness analysis seeks (new and) robust solutions.

Robustness analysis comes, broadly speaking, in two forms. In robustness under uncertainty analysis, we specify worst case developments (e.g., changes in θ_is or in e_is that are credible and that would adversely affect a favored position), and we seek best meliorating responses to them. In robustness under risk analysis, we specify probability distributions on model components and then seek solutions that meet a probabilistic performance standard (e.g., with probability of at least x will yield a result of at least y).

- *Robustness analysis has been explored principally in the context of constrained optimization models* **??**. *It is also apt in our context of modeling for the group decision forecast problem.*
- *Individual exploration services, as illustrated briefly in §7.2.5, afford robustness under uncertainty analysis.*
- *Group sampling exploration services, as illustrated briefly in §7.2.5, afford robustness under risk analysis.*

Model structure analysis is a fourth kind of post-solution analysis. It mainly considers alternative modules for a given model or alternative models entirely. In undertaking model structure analysis each of the aforediscussed forms of post-solution analysis pertain.

- *Our exploration of several different models for the case supports an elementary form of model structure analysis.*

7.2.5 Post-Solution Analysis: Services and Techniques

We have implemented models 1 and 2 in the NetLogo model IranBargaining.nlogo. This (computational model, a computer program) including the data file we have discussed above may be found at the NetLogo Modeling Commons: `http://modelingcommons.org` (search on "IranBargaining").

Instructions for using the IranBargaining model are given in the Info tab of the NetLogo model. Here, we focus on the using the model to illustrate

concepts. Figure 7.4 shows the Interface tab of the IranBargaining NetLogo model after execution of model 1. Instructions for using the IranBargaining model are given in the Info tab of the NetLogo model. Here, we focus on the using the model to illustrate concepts.

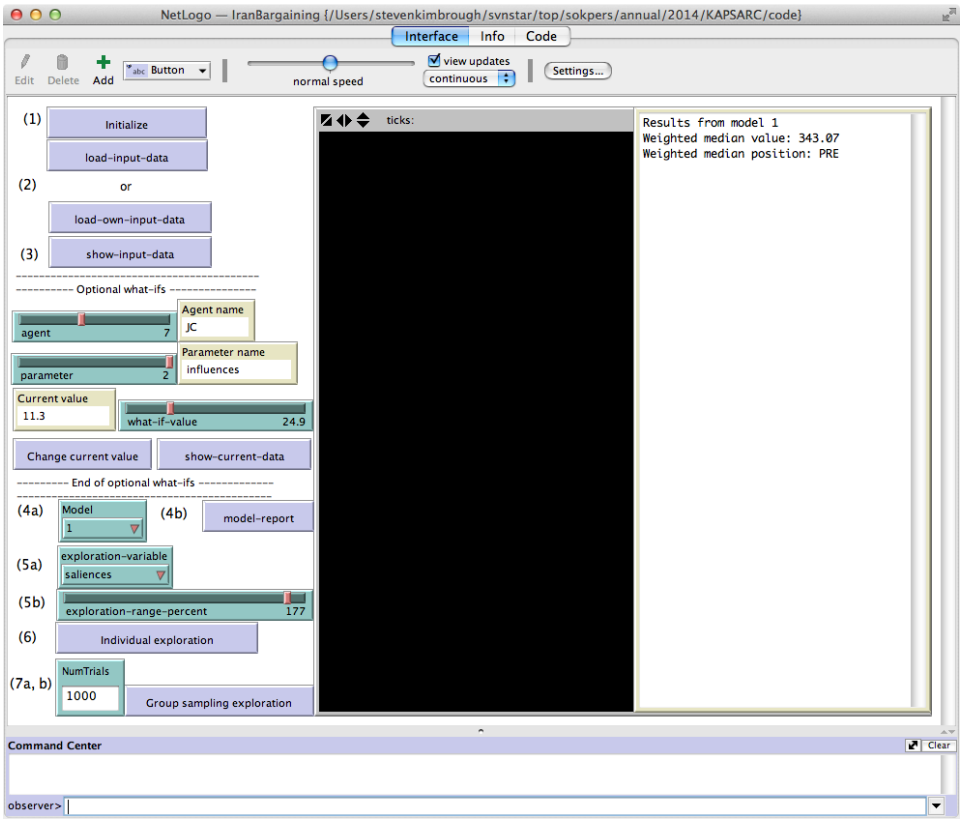

Figure 7.4: IranBargaining NetLogo model, results from executing model 1.

7.2.5.1 What-If Analysis

The IranBargaining model implements three very general capabilities for post-solution analysis. *What-if* is the first of these capabilities. Using the what-if features of the NetLogo model the user may change one or more parameter values and then execute an associated model to see what the effect is of the change(s). Figure 7.5 shows the Interface tab of the IranBargaining NetLogo model after changing the influences value of agent JC

from its default value of 11.3 to 24.9 and executing model 1. We see (in the upper right-hand area of the display) that the position predicted of the group changed from PRE to JC. The increase in JC's influence has been sufficient to alter the outcome of the group's deliberations, according to the model.

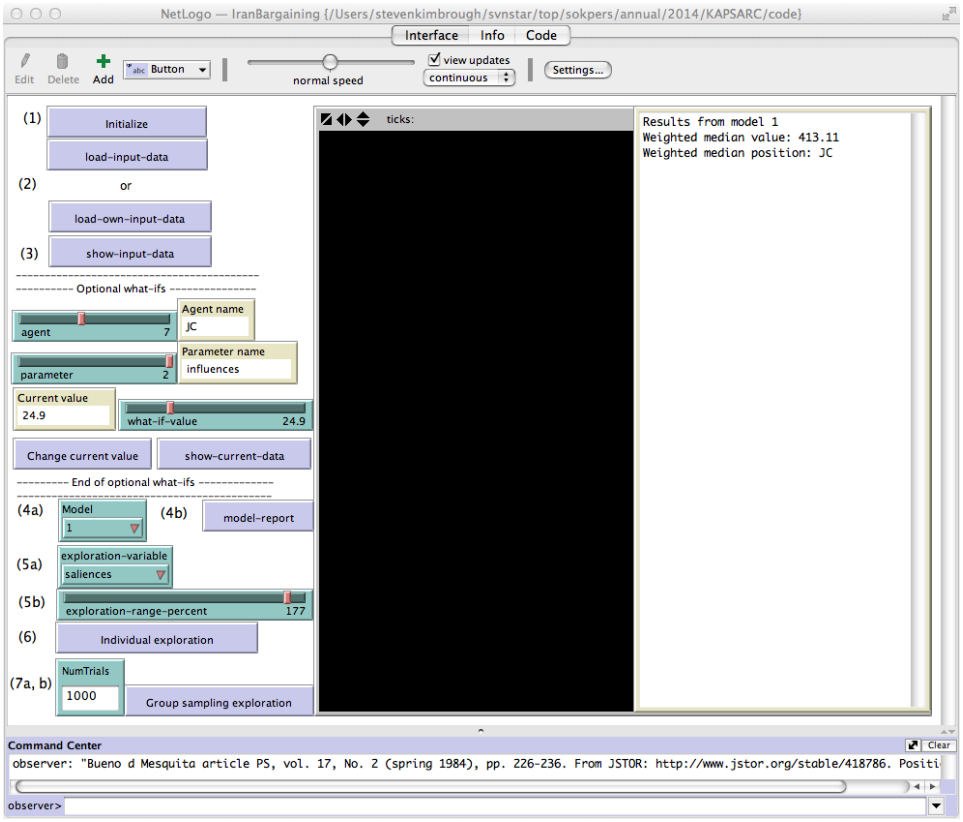

Figure 7.5: IranBargaining NetLogo model, results from executing model 1 after changing the influences value for agent JC from its default value of 11.3 to 24.9.

If, for example, we change the influence value of agent JC from its default value of 11.3 to 24.9 and execute model 1, we will see that the position predicted of the group changed from PRE to JC. The increase in JC's influence has been sufficient to alter the outcome of the group's deliberations, according to the model.

7.2.5.2 Individual Exploration

A form of individual exploration analysis is supported in the IranBargaining Netlogo model with the code associated with the "Individual exploration" button, labeled (6) in the display. Figure 7.6 shows the Interface tab of the IranBargaining NetLogo model after executing individual exploration on model 1 with an exploration range of 177 percent on the salience values for the 18 agents. The meaning of the output, shown in white lettering on a black background in the figure, is as follows. The left-hand column of the output arrays the short names of the 18 agents in the data for model 1. The order is in their position values, increasing downwards. So UMC has the smallest position value and QUM the largest. The values in the right-hand column are the decisions reached by the model when the corresponding agent in the left-hand column has its salience score enlarged by 177 percent. For example, for REV in the left-hand column, we find JC in the right-hand column. This means that if we keep the input data (perhaps as changed by what-if operations) constant, but increase the salience value for REV by 177 percent, then the group is predicted to resolve its decision by choosing the JC position. If, for example, we set the exploration range to 177 percent on the salience values for the 18 agents we get a report whose meaning is as follows. For each of the 18 positions we see the decision reached by the model when the corresponding agent has its salience score enlarged by 177 percent. For example, for REV, we find JC is the new decision. This means that if we keep the input data (perhaps as changed by what-if operations) constant, but increase the salience value for REV by 177 percent, then the group is predicted to resolve its decision by choosing the JC position.

Overall, what the NetLogo model model is telling us here is that even with such a large change PRE still occurs most often. However, there are several cases in which it is altered. Note that the output reported contains information from 18 distinct solutions of model 1. This is valuable not only for saving analyst time by avoiding a one-a-time what-if analysis; it also presents a higher-level pattern for viewing by the analyst and decision makers. This is a simple instance of application of the principle of *solution pluralism* for post-solution analysis.

7.2.5.3 Group Sampling Exploration

In individual exploration we systematically make changes to one variable at a time and observe the behavior of the model. In group sampling explo-

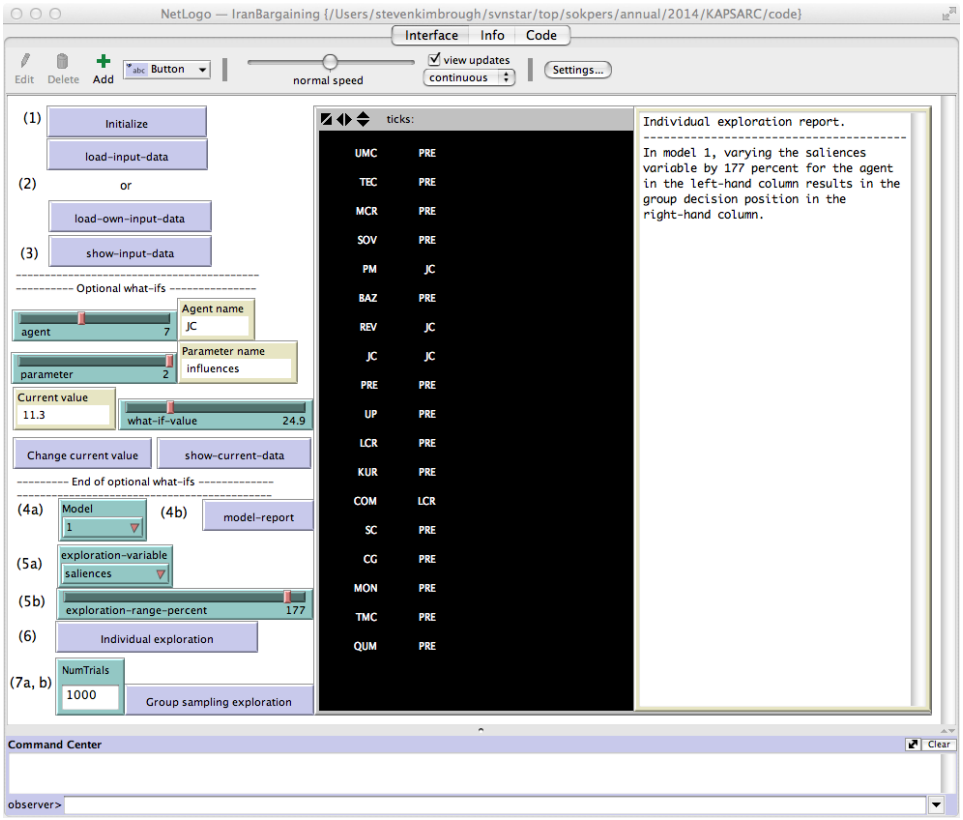

Figure 7.6: IranBargaining NetLogo model, results from individual exploration on model 1 using the default parameter settings and an exploration range of 177 percent on salience values.

ration we randomly change a number of variables at once and record the behavior of the model. We repeat this process a large number of times and observe the distribution or pattern of behavior of the model.

A form of group sampling exploration analysis is supported in the IranBargaining NetLogo model with the code associated with the "Group sampling exploration" button, labeled (7b) in the display. Figure 7.7 shows the Interface tab of the IranBargaining NetLogo model after executing group sampling exploration on model 1 with an exploration range of ± 75 percent on the salience values for the 18 agents. For example, if we set the exploration range to ± 75 percent on the salience values for the 18 agents we can get a report of the consequences.

The meaning of the report, shown in white lettering on a black back-

ground in the figure, is as follows. The left-hand column of the output arrays the short names of the 18 agents in the data for model 1. The order is in their position values, increasing downwards. So UMC has the smallest position value and QUM the largest. The values in the right-hand column are the number of times the decisions reached by the model for the corresponding agent position in the left-hand column. This is under perturbation of the salience values in which the salience value for every agent is randomly perturbed in its ± 75 percent range and model 1 is executed to predict an outcome. For example, for REV in the left-hand column, we find 2 in the right-hand column. This means that in 10,000 trials (see "NumTrials" on the Interface, labeled (7a)) exactly 2 produced joint salience values resulting in a predicted position of REV as the outcome.

Overall, what the NetLogo model model is telling us here is that even with such a large change PRE still occurs about 70 percent of the time. However, there is definitely a distribution of outcomes. JC will occur about 20 percent of the time and the other outcomes are concentrated on the high side (lower in the display) of the PRE position (i.e., favoring vigorously pursuing the war).

We see again that a post-solution analysis capability is valuable not only for saving analyst time by avoiding a one-a-time what-if analysis; it also presents a higher-level pattern for viewing by the analyst and decision makers. This is another simple instance of application of the principle of *solution pluralism* for post-solution analysis.

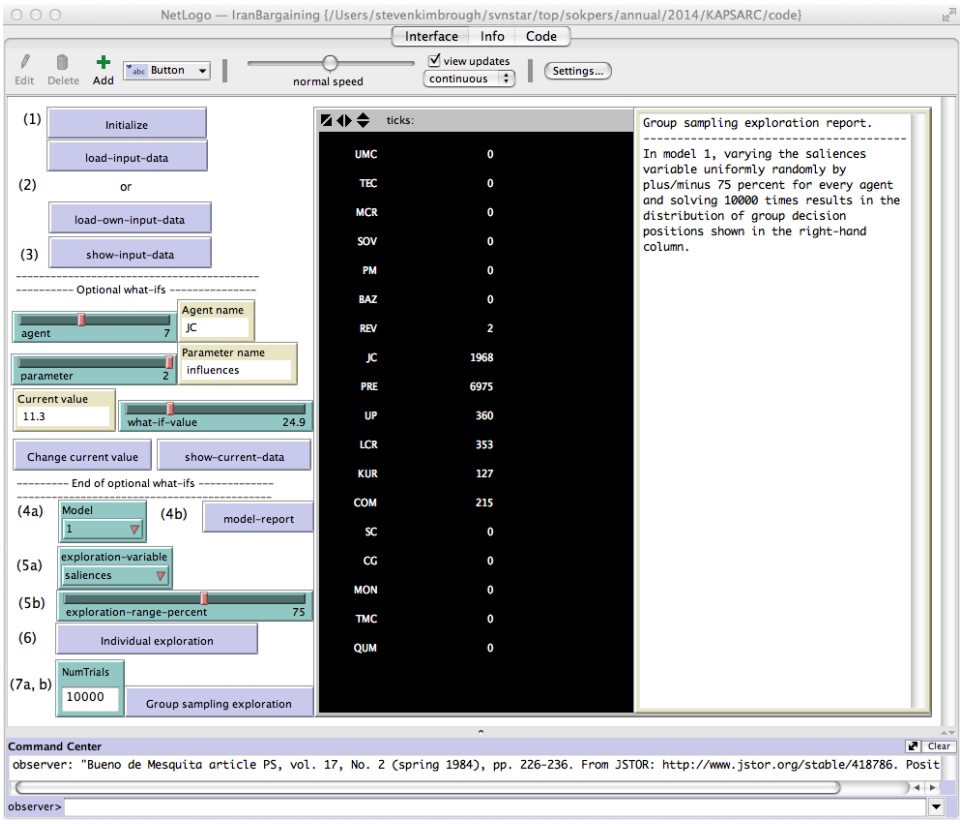

Figure 7.7: IranBargaining NetLogo model, results from group sampling exploration on model 1 using the default parameter settings and an exploration range of ± 75 percent on salience values.

7.3 Discussion

There are two principal themes or topics in this chapter. First, we introduced position bargaining models for group decision forecasting (GDF) problems. These models are designed to apply in GDF problems in which the group is deliberating about which position along a single dimension it should take. This is the simplest of a broad range of GDF problems, but those with multiple issues, or multiple dimensions of interest, are perhaps the most important of the more complex problems. The four models we described are examples of generalized voting model, in which the forecast for the group is arrived at by assuming a form of voting, selected from the many that are available **?**. It is to be emphasized that the modeling exercise does *not* assume that the group in question

Second, we introduced post-solution analysis and the principle of solution pluralism for doing it. These are themes that will recur throughout what follows.

7.4 For Exploration

7.4.1 Non-Programming Exercises

1. Find and analyze a GDF problem of your own.
2. Each of the four generalized voting models we discussed has the property that it forecasts adoption by the group of one (or more) of the positions taken by the members of the group. This forecloses the possibility that the group will arrive at an outcome intermediate between two of the positions of its members. Is this realistically a defensible property for these models to have? Discuss the pros and cons.
3. Discuss: How valid is Bueno de Mesquita's approach, outlined here? How would you go about testing and evaluating it? How might it be improved?
4. Read **?** and assess how accurate the predictions in it are.
5. Discuss other applications of generalized voting models for GDF problems. How accurate have they been? What are some successes? Failures?
6. Investigate alternative models for our use case of single dimensional bargaining.

7. Document one or more real cases of GDF problems, including examples that are more complex than out single dimension case.

8. Investigate models for more complex GDF problems.

7.4.2 Programmin Exercises

1. Implement model 3 in the IranBargaining NetLogo model.

2. Implement model 4 in the IranBargaining NetLogo model.

3. Implement models 1, 2, 3, and 4 in Python.

7.5 For More Information

See **?** for a recent description by Bueno de Mesquita of this approach and methods. See **?** for a recent, dogged attempt to ascertain the actual model(s) used by Bueno de Mesquita for predicting group decisions.

The *Median Voter Model* (MVM), which has its *Median Voter Theorem* (MVT), is related to our discussion and comes with its own large literature. The model is normally associated with **?** who worked with spatial models of economic competition, with **?** who formalized its necessary and sufficient conditions and who placed it in voting contexts, and with **?** whose book developed a very general and wide-in-scope approach. For those who enjoy animated presentations of such results, see `http://www.youtube.com/watch?v=cFt0k6n_HKc`. Further, there is in fact considerable empirical support for the MVM. The following passage is representative of the literature.

> Although theoretical arguments suggest that the applicability of the median voter model may be very limited, the empirical evidence suggests otherwise. There is a large body of evidence that suggests median voter preferences over policies are (largely) of the sort which can be mapped into a single issue space while retaining "single peakednes". **?** find that 80–90% of all the recorded votes in the US Congress can be explained with a one dimensional policy space. **?** finds little evidence of cycling across Congressional votes over district specific grants. **?**

7.6 Acknowledgments

Support for this work was provided in part by KAPSARC (`http://www.kapsarc.org`). Brian Efrid, Leo R. Lester, and Ben Wise were instrumental in providing background information, comments, and suggestions.

Part III

Exploratory Modeling

Appendix A

Development Notes

A.1 2015-09-13

So, I've begun working on the Teaching Notes for Thinking with Models (DOID 325) for spring 2016. The immediate goal is to get ready to teach the class in spring, especially with regard to in-class exercises, and notes for them. I plan to do this, focusing on the chapters (one per class) in Part I of the Notes.

The intermediate goal is to design and work towards a new book, called *Thinking with Models*, that I can use for the class and that teaches, especially, solution pluralism and exploratory modeling. Puzzled a little about organization. One thing also is that I'm ambivalent about the longer term use of NetLogo. Why not just switch to Python? Well, there's a very nice NetLogo library and a lot of students chary of programming might find this ok. So, I'm thinking of doing follow on graphics and data analysis in Python, and maybe some models. Maybe something like this:

Part I Programming NetLogo.

Keyed initially to the Wilensky and Rand book. Certainly do this for spring 2016 (i.e., now!).

Part II Modeling examples.

Provide an early chapter surveying the field; kinds of models, drawing upon distinctions in chapter 1.

Already: Converse, Huff, Bueno de Mesquita. Coming: Solar PV Adoption, Ambidexterity Strategy Explorer, Probe and Adjust for Electricity Bidding. Probably: Models with Robin Clark.

What else? Could be Python models, even MATLAB models. Want to have models that afford deliberation and that connect with real decision making.

Here, I could use material from the BizAnalyticsBook with HC. Chapters here and in Part II.

Part III Exploratory modeling.

These chapters might be matched with chapters in Part II. Here I discuss exercising the models, solution pluralism, exploratory modeling, and I do lots of data analysis and graphics, all in Python.

Well, this seems ok. Now to it! Note that there will be an accompanying document, *In-Class Exercises*.