GEOG 5160/6160 Lab 07

Simon Brewer

1/27/2018

## Lab 07: Uncertainty analysis

## Introduction

In this lab, we will look at tools for assessing model uncertainty in Netlogo.

## BehaviorSpace

Netlogo comes packaged with a set of tools that extend its basic capabilities. On of these tools, BehaviorSpace, is designed to help setup and run large series of experiments with a model. This allows you test the the effect of varying one or more parameters systematically, recording the model outcome each time. In addition, BehaviorSpace allows repeated runs of a model, testing the impact of stochasticity through examining the range of outcomes with different random starts (e.g. different random initialization of the landscape).

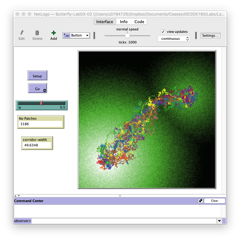
You have done some similar work in previous labs, where the exercise has asked you to run models with different settings, but this quickly becomes impractical. For example, a model that has three parameters, each with 10 possible settings, has possible combinations to run. If, in addition, we want to run each combination 100 times to check for the impact of stochasticity, then this would require 100,000 model runs.

Running a model multiple times is a relatively straightforward parallel problem in computing, and BehaviorSpace can take advantage of multiple processor cores, if these are available.

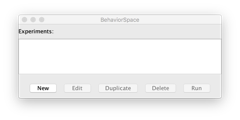
## The butterfly model

We’ll start by repeating the experiment you ran with the Butterfly model. We’ll use the original model with two hills, and 50 butterflies. You can use your own model, or the model *Butterfly-Lab03-02.nlogo*, which should be available through Canvas. As a reminder, this model has a single

Prior to setting up experiments in BehaviorSpace, open this model, check the code and do a test run with to make sure everything is working. The results should look similar to the plot below.



Start BehaviorSpace from the [Tools] menu, and select a new experiment from the following window. The BehaviorSpace experiments you set up can be saved and reopened here later.



Clicking on [New] will open the following window, in which the experimental options can be set.  


There are a lot of options here, so we’ll start with a simple model run to test the effect of stochasticity, with set to 0.5. Set the following options:

1. Experiment name “butterfly-q-0.5”. The experiment will be stored with this name, and this will be used to name all output files
2. Repetitions: 100, to run 100 times with different random starts
3. In the next box titled “Measure runs using these reporters”, add “corridor-width”. This is the variable that will be output from the model from each run. Here, we use corridor width as it is already calculated in the model, but you can also enter Netlogo code to output other variables. Also note that several variables can be recorded.
4. Uncheck the box “Measure runs at each time step”. If checked, this will output a value at each time step for each repetition of each experiment. This can be useful, but here we are only interested in the final corridor width.

At the bottom of the window are various options to setup and stop the model. Stopping the model is, obviously, very important, as we don’t want to risk models running to infinity. As we already have set a time limit (1000 ticks) in the code, we do not need to add anything. Leave everything else as it is, click on [OK], and you should now see the model name appear in the list of BehaviorSpace experiments. Highlight your experiment and click [Run]. Note that if you need to change anything, then the [Edit] button will reopen the high-lit experiment.

A new window will open asking if you want spreadsheet or table output, check both of these boxes. The table output is more useful, but it is worth at this point seeing the difference between them. This window will also display the number of parallel simulations to be run. This will be set by default to the number of processor cores that your computer has. It is probably best to leave this number unchanged, (unless you desperately need to do something else on your computer).

Click [OK] and you will be prompted for a location to save the output files, and then the experiment will start. This took about 4 seconds on my aging Macbook Pro, but will vary depending on the computer you use and the cores available. You may want to uncheck the boxes “Update view”" and “Update plots and monitors”. This will stop all graphic output and will speed up the model.

Once the experiment has run, open the “table” file that was created. This is a csv-file and should open directly in Excel. The first few lines of the file give some information about the model run (the model name, the data and time, etc). From line 7 onwards are the results: - Column 1: the run number - Column 2: the value of q - Column 3: the number of ticks in the run - Column 4: the final corridor width

(If you open the “spreadsheet” file, you will see it is the same thing, but transposed.)

We’ll now use R to read and visualize the output (you are welcome to use other graphing software, but R integrates well with Netlogo). Start R (or preferably RStudio), and change the working directory to the folder where the Netlogo output files are. The working directory can be set from the [File] menu on Windows, the [Misc] menu on a Mac, or the [Session] menu if you are using RStudio.

Now check that the files are in the directory by typing the following the R console window:

list.files(pattern=".csv")

Now load the file into a R data frame (called dat) using the read.csv() function. Note that we set the parameter skip=6 to avoid reading in the first six lines of metadata.

dat = read.csv("Butterfly-Lab03-02 butterfly-q-0.5-table.csv", skip=6)  
names(dat)

## [1] "X.run.number." "q" "X.step." "corridor.width"

Calculate summary statistics on the corridor widths (which suggest a slight left skew):

summary(dat$corridor.width)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 37.39 47.86 52.55 52.08 57.26 62.63

And now plot a histogram. You can do this using base R as follows:

hist(dat$corridor.width, main="Corridor width (q=0.5)")

or by using **ggplot2**. I would recommend using this as a) it makes nicer looking graphics and b) it will make some plots easier. Note that **ggplot2** is an add-on library, and you may first need to install it:

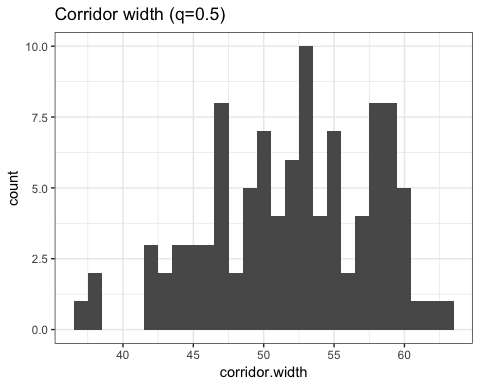
install.packages("ggplot2")

To make a **ggplot2** histogram, first load the library:

library(ggplot2)

Then make the histogram

qhist = ggplot(dat, aes(x=corridor.width)) + geom\_histogram(binwidth = 1)  
qhist = qhist + ggtitle("Corridor width (q=0.5)") + theme\_bw()  
print(qhist)



### Varying parameter settings

Now return to Netlogo. We will now test the effect of changing the values of in the model. Reopen the BehaviorSpace window ([Tools]->[BehaviorSpace](#behaviorspace)), and make a duplicate of the existing experiment. In the experimental options:

1. Call this new experiment ‘butterfly-q’
2. In the “Vary variables as follows…” window, set a range of values of to test. We will vary this from 0 to 1 by steps of 0.1. Variables have to be entered in brackets, with the values to be test, so ‘[“q” 0.1 0.4 0.9]’ would run tests with 3 different values of . For our experiment, we can give a minimum, increment and maximum as follows (note the extra brackets): ‘[“q” [ 0 0.1 1 ]]’

We’ll keep all other options as before. Now, click on [OK], and you should now see the new model name appear in the list of BehaviorSpace experiments. Highlight your experiment and click [Run]. In the next window make sure that ‘table’ output is checked (you can uncheck the ‘spreadsheet’), and click [OK]. Remember to uncheck the “Update” boxes to get your model to run faster. In the monitor window you should see the value of increasing across the various model runs.

When it is done, read the new table file into R.

dat = read.csv("Butterfly-Lab03-02 butterfly-q-table.csv", skip=6)

As there are now multiple values of , we need to calculate summary statistics for each one. R has a few approaches to doing this, but here we will use a package called **dplyr**. Install this first, if it is not already present on your computer.

install.packages("dplyr")

Now load the library:

library(dplyr)

And calculate the group means and standard deviations for each value of . This should be fairly straightforward; the data frame dat is split into groups by the value of q, and the summarize function creates two new variables, one with the mean, one with the s.d.

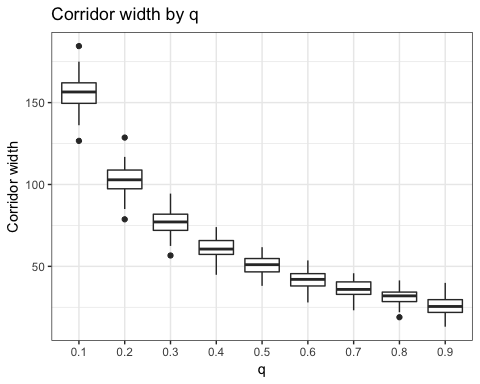
dat %>%   
 group\_by(q) %>%  
 summarize(avg\_cw = mean(corridor.width),  
 sd\_cw = sd(corridor.width))

If we save this new summarized dataset, we can then use it to look at the results

dat.sum <- dat %>%   
 group\_by(q) %>%  
 summarize(avg\_cw = mean(corridor.width),  
 sd\_cw = sd(corridor.width))  
  
myplot <- ggplot(dat.sum, aes(x=q, y=avg\_cw)) + geom\_line()  
print(myplot)  
  
myplot <- ggplot(dat.sum, aes(x=q, y=sd\_cw)) + geom\_line()  
print(myplot)

As the value of increases, this lowers the chance of a butterfly moving in a random direction, and this is reflected in a narrower corridor and a reduction in variance. A better way of looking at this is with a box-plot, where the box and whiskers reflect the range of outcomes for each value of .

dat$q <- as.factor(dat$q)  
myplot <- ggplot(dat, aes(x=q, y=corridor.width)) + geom\_boxplot()  
myplot <- myplot + ggtitle("Corridor width by q") + theme\_bw() +  
 scale\_y\_continuous("Corridor width")  
print(myplot)



### Multiple parameters

In this next section, we test the effect of varying multiple parameters in an uncertainty analysis. For this we use the model of fire spread introduced in lab 05. To ensure that we are all using the same version of this model, download and use *FireModel-Lab07.nlogo* from Canvas. This is the second iteration of this model with settings for density, ignition and extinction, but without the variation in ignition probabilities to represent wind effects. Three extra modifications have been made:

* A slider has been added to control isotropic ignition probabilities.
* Fires now start from a randomly selected patch with vegetation, rather than the center
* The model stops when all fires have been extinguished

As before, open the model and run a single simulation to check that everything works.

For this experiment, we are going to explore different combinations of the three free parameters (density, ignition and extinction). The main outcome variable is already defined in the model, and gives the percent of vegetation that has burned (percent-burned), but we will add a second variable giving the number of patches burned. Set the following options for the BehaviorSpace run:

1. Experiment name: ‘fire-all’
2. Variables to vary:
   * [“density” [10 10 90]]
   * [“extinction” [0.1 0.1 0.9]]
   * [“ignition” [0.4 0.05 1]]
3. Number of repetitions: 10
4. Measure runs using:
   * ‘percent-burned’
   * ‘count patches with [ pcolor = 12 ]’
5. Uncheck ‘Measure runs at each step’

Click [Ok] to save the experiment, then select it from the main BehaviorSpace menu and [Run]. Remember to select ‘table’ output, and uncheck the ‘View’ options once this starts. This will require about 10,000 model runs, and there will be a large variation in the time it takes models to run, as there is no hard time limit.

When it is finished, read the data into R as before:

dat <- read.csv("FireModel-Lab07-2 fireExp2-table.csv", skip=6)  
names(dat)

Let’s start by plotting the response to the individual parameter settings:

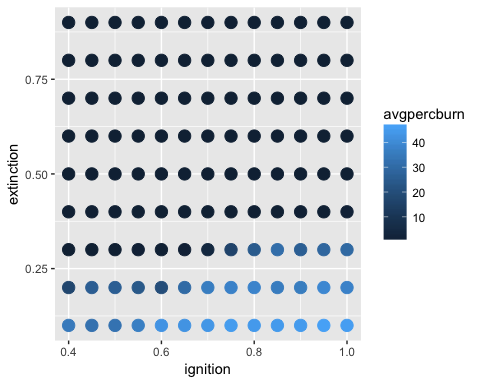
myplot <- ggplot(dat, aes(x=factor(density), y=percent.burned)) + geom\_boxplot()  
myplot <- myplot + ggtitle("Percent burned by d") + theme\_bw() +  
 scale\_x\_discrete("Density") + scale\_y\_continuous("%")  
print(myplot)  
  
myplot <- ggplot(dat, aes(x=factor(ignition), y=percent.burned)) + geom\_boxplot()  
myplot <- myplot + ggtitle("Percent burned by ignition") + theme\_bw() +  
 scale\_x\_discrete("Ignition") + scale\_y\_continuous("%")  
print(myplot)  
  
myplot <- ggplot(dat, aes(x=factor(extinction), y=percent.burned)) + geom\_boxplot()  
myplot <- myplot + ggtitle("Percent burned by extinction") + theme\_bw() +  
 scale\_x\_discrete("Extinction") + scale\_y\_continuous("%")  
print(myplot)

While these show roughly what we would expect (i.e. higher percent burned when density is high, ignition is high and extinction is low), some of the pattern is difficult to make out as these include the interactions between pairs of parameters. We can do this in two steps. First, make a temporary data set with the average percentage burned for each combination of parameters:

dat2 <- dat %>%  
 group\_by(ignition,extinction) %>%   
 summarize(avgpercburn = mean(percent.burned))

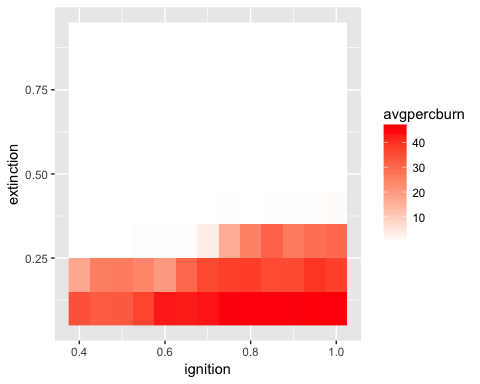
Then plot this as a scatterplot, which shows a clear peak at high values of ignition and low values of extinction:

myplot <- ggplot(dat2, aes(x=ignition, y=extinction, col=avgpercburn)) +   
 geom\_point(size=4)  
print(myplot)



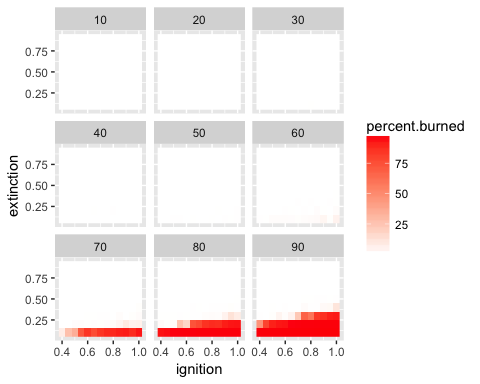
An alternative visualization can be made with the geom\_raster function, which yields a continuous surface:

myplot <- ggplot(dat2, aes(x=ignition, y=extinction, fill=avgpercburn)) +   
 geom\_raster() + scale\_fill\_continuous(low="white", high="red")  
print(myplot)



We can take advantage of **gplot2**’s facet\_wrap function, to see how this pattern varies with different values of density. Note that this uses the original dataset:

myplot <- ggplot(dat, aes(x=ignition, y=extinction, fill=percent.burned)) +   
 geom\_raster() + scale\_fill\_continuous(low="white", high="red") + facet\_wrap(~density)  
print(myplot)



## Exercise

Use a word document to record your answers and output. Assignments, to include both the word document and .nlogo script file for the model, should be submitted to Canvas before class on Monday, XXXXXX XX. Ensure your assignment has been saved using the following naming convention: Lab06\_lastname\_script and Lab06\_lastname\_report.

1. Using the code given above, make plots to show how the size of the fire and the duration of fire vary with different values of density, ignition and extinction probabilities. Write a short paragrah to describe what you see [2]
2. In lab 04, you built a foraging model and ran some simple tests to look at how the five parameters of the model impacted the simulation time (the number of ticks). Choose three of these five parameters and design a BehaviorSpace experiment to test the the impact of changing these in a systematic way. For the remaining two parameters, use the default values from the list below. You will need to describe the experiment: which parameters you choose and why, what values were used, and how many repetitions you used, as well as figures showing the resulting simulation time [3]. As it is possible that certain combinations of parameters will lead to simulations that do not stop, add the following stopping rule to the go procedure:

if ticks > 50000 [ stop ]

Foraging model parameters