Agent-Based Modelling in NetLogo

Experimental methods and tools

David Hales

www.davidhales.com/abm-netlogo

Verification, Validation, Calibration

- Verification:
  - Checking the model is a correct implementation of what was intended (the specification)
  - No bugs or logical errors
- Validation:
  - Checking model against empirical data (often using "**statistical signatures**" or "**stylised facts**" or qualitative evaluations)
- Calibration:
  - Setting various parameters of model based on empirical data

Verification

- Specifications often given at the agent level using natural language, formulae, diagrams etc. (a "**conceptual model**")
- Since we generally don't know what will happen when we run the model (emergence):
  - Verification can be helped by independent implementation and **replication** from the conceptual model
    - If a replication fails then either the conceptual model is unclear and / or one or both of the implementations is incorrect
    - In general the more independent replications of a model the more confident we become that the models are valid implementations

Edmonds, B. and Hales, D. (2003) Replication, Replication and Replication - some hard lessons from model alignment. Journal of Artificial Societies and Social Simulation 6(4) <http://jasss.soc.surrey.ac.uk/6/4/11.html>

Validation

- What constitutes validation of ABM's is controversial (particularly for social systems):
  - quantitative prediction of target systems?
  - qualitative stylised facts of target systems?
  - issues of future & past data and over-fitting
  - validate agent behaviour & emergent outcomes?
  - validation as "model is useful to users" ?

example model / validation

- Sample models > Networks > Team Assembly
- Implementation of a dynamic network ABM that relates to a theory of creative team formation
- Related to empirical data (plays, movies, scientific papers)
- How convincing is this work as a form of validation of an ABM?

Paper: R Guimera, B Uzzi, J Spiro, L Amaral (2005) Team Assembly Mechanisms Determine Collaboration Network Structure and Team Performance. Science, 308(5722), p697-702

Models as thought experiments

- Many ABM models are not validated:
  - there may be no target system or do not predict a target system or relate to data
  - thought experiments: implications of assumptions, theory development (pure)
  - Construction: aid engineering of systems (applied)
  - Possible worlds: artificial life / artificial societies
  - "fact free science" or "a 3rd way of doing science"?

Epstein, J. (2008) Why Model? Journal of Artificial Societies and Social Simulation vol. 11, no. 4. http://jasss.soc.surrey.ac.uk/11/4/12.html

Axelrod, R. (1997) Advancing the art of simulation in the social sciences. R. Conte and R. Hegselmann (eds) Simulating Social Phenomena. Springer.

KISS and KIDS

- There are different approaches to formulating an ABM:
  - "keep it simple stupid" (KISS): make the model as simple as possible to start with then add bits
  - "keep it descriptive stupid" (KIDS): make the model as descriptive as possible then remove bits

Edmonds B and Moss S (2005) From KISS to KIDS — an 'anti-simplistic' modelling approach. In Davidsson P et al (eds.) Multi Agent Based Simulation. Springer.

Combining Induction & Deduction

- ABM methods combine (in different ways):
  - Deduction: the simulation model executes a computer program performing deductions from assumptions
  - Induction: the experimenter observes the model runs and attempts to identify patterns and relationships
- A simple form of this is the "existence proof": a given ABM is sufficient to produce some result

Hales, D., (2010) Mix, Chain and Replicate: Methodologies for Agent-Based Modelling of Social Systems. In Mollona, E., (ed) Computational Analysis of Firms' Organization and Strategic Behaviour. London and New York: Routledge

Exploring a model

- Given some ABM how do we go about exploring what it does?
- Most models will have:
  - parameters that can be set over some range
  - some output measures
- We can perform “experiments”:
  - execute sets of simulation runs for different parameter values
  - collect the output measures

Experiments with models

- Given an ABM model we can:
  - generate data by running experiments
  - control the input parameters
  - analyse the data using any statistical or analytic method we wish (even machine learning etc)
  - treat model like a science experiment in a lab
  - have hypotheses about what experiments will produce (hunches)
  - Form explanations of how agent behaviours produce results observed (generate theory)

An example experiment

- A simple **sensitivity analysis**:
  - Vary a single parameter over some range
  - For each value perform a number of simulation runs with different initial random number seeds
  - Each run executes for some number of cycles or until some stopping condition
  - Collect output measures for each run
  - Graph the results to see how the input parameter effects the output measures

NetLogo BehaviorSpace tool

- NetLogo has tool called “BehaviorSpace” that automates experiments of systematic sets of runs, writing results to a CSV file
- Can load CSV into other statistical applications for visualisation and analysis
- BehaviorSpace can use all the processor cores on the machine, doing runs in parallel
- BehaviorSpace can be run in the background without the netlogo display (headless)

Example: Segregation experiment

- Use BehaviorSpace tool to do an experiment to explore the Schelling segregation model
  - Sample models / social science / segregation
- Model has two input parameters: “density” and “%similar-wanted”
- We will explore the effect of varying the %similar-wanted parameter on the output measure percent-similar (a segregation measure) and time to reach stability (ticks)

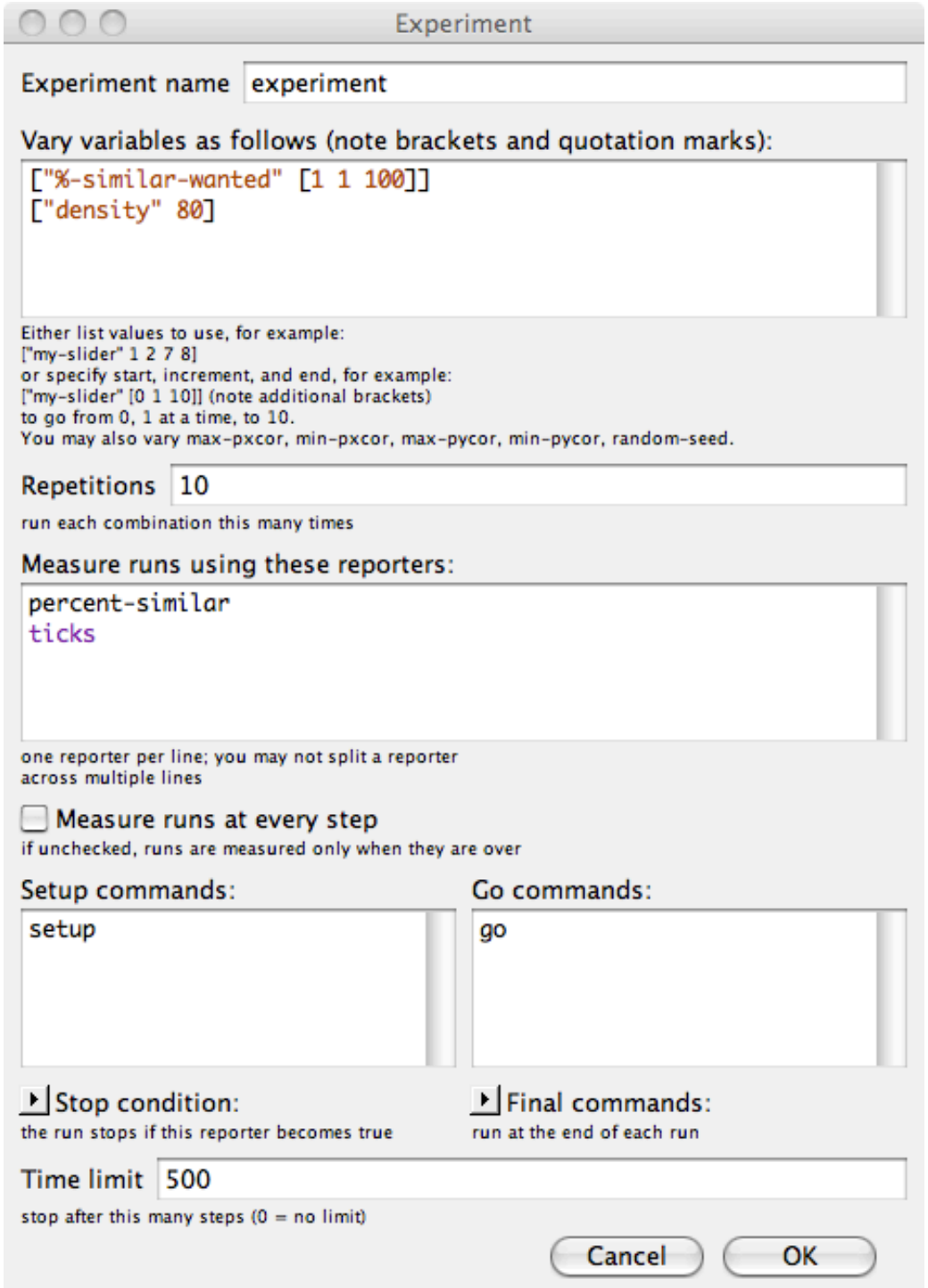
Segregation experiment

- Since the model is not deterministic (it uses random numbers) it is wise to perform a number of runs for each value of %similar-wanted
- Since runs with high values of %similar-wanted never stop we need to put a cut-off at some number of steps
- We will do a scan of:
  - %similar-wanted values from 1..100 in increments of 1
  - With density held constant at 80
  - For each %similar-wanted value we will do 10 runs
  - Terminate runs at tick 500 if it has not already terminated
  - Report the percent-similar output measure (at end of each run) and number of ticks each run took

A NetLogo “experiment” defined under Tools>BehaviorSpace for the Segregation model.

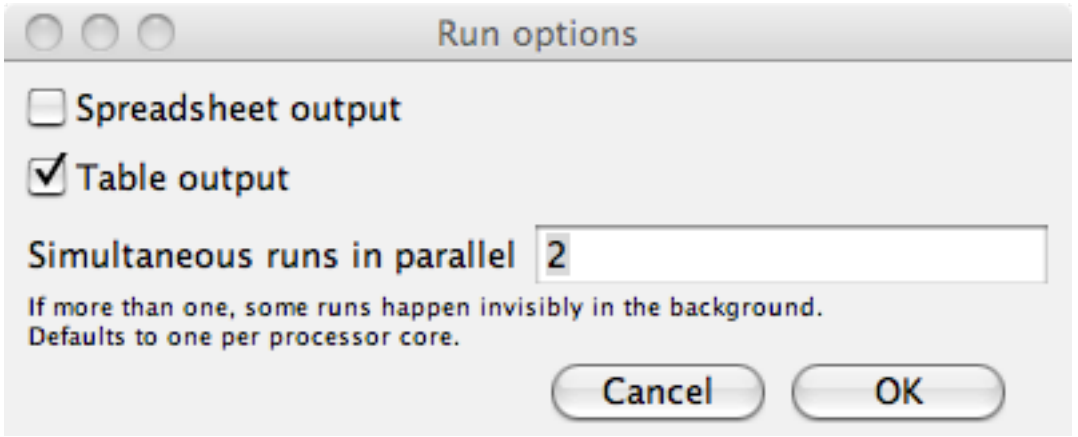
It takes quite a while to run on a standard laptop (NetLogo is slow) but can take advantage of multiple cores on bigger machines

Since NetLogo is written in Java you can easily run experiments on servers without recompiling. See “headless” slides later



Running experiment

Selecting an experiment from BehaviourSpace and clicking Run will bring up the run options dialogue box =>



Here we select Table output which gives a simple CSV file in the form of one line of the file for each run

Since this machine has two processor cores then the default is two runs in parallel (one on each core)

Results file

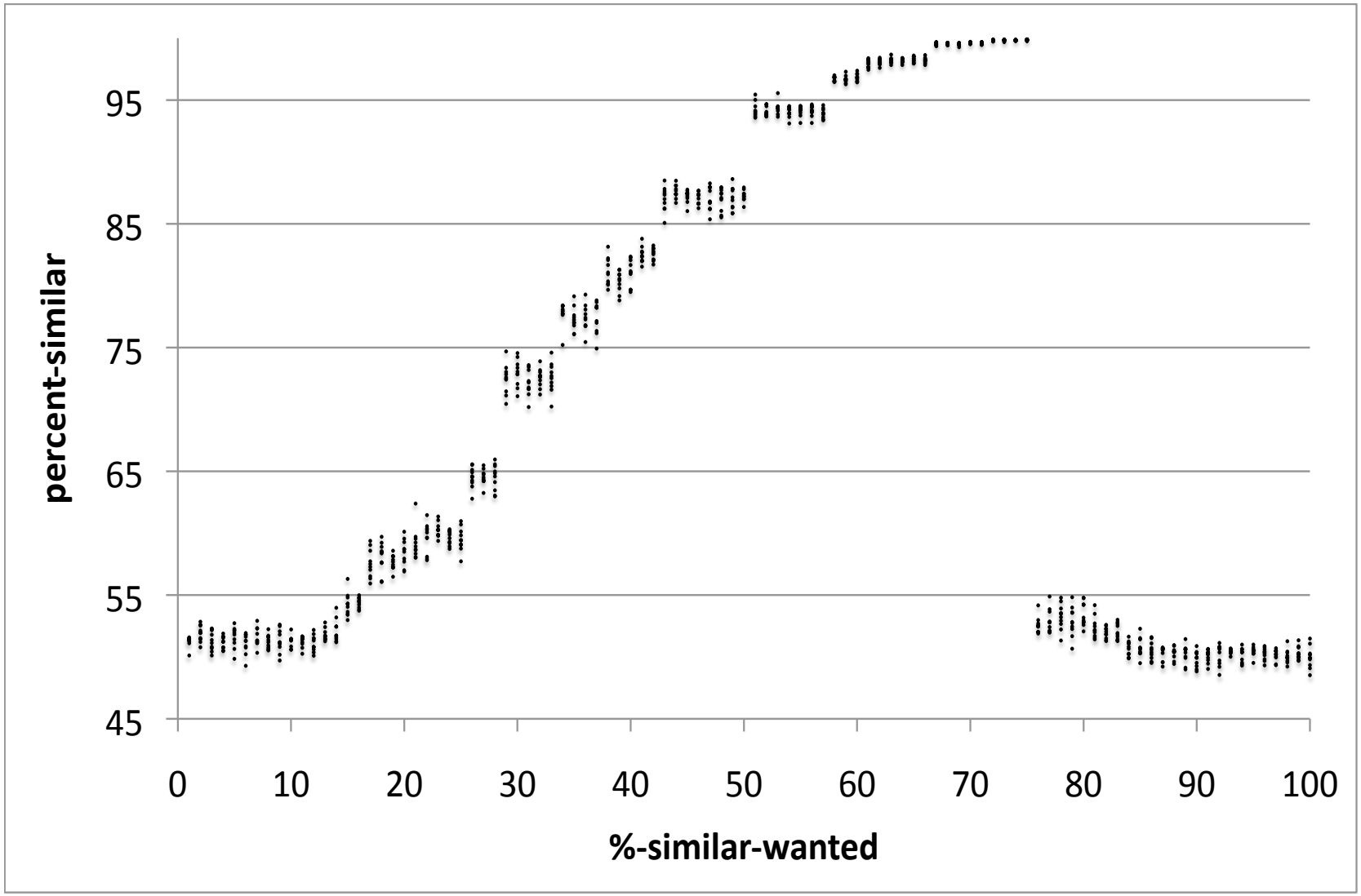
Should get a CSV output file that looks something like this:

```
"BehaviorSpace results (NetLogo 5.2.0)"
"seg.nlogo"
"experiment"
"05/01/2016 13:12:24:026 +0200"
"min-pxcor","max-pxcor","min-pycor","max-pycor"
"-25","25","-25","25"
"[run number]","%-similar-wanted","density","[step]","percent-similar","ticks"
"2","1","80","2","52.26993865030675","2"
"3","1","80","2","50.683696468820436","2"
"1","1","80","2","50.21625652498136","2"
"4","1","80","2","51.13721946076216","2"
"6","1","80","3","51.47405660377359","3"
"7","1","80","3","52.02985074626866","3"
"5","1","80","2","52.21198852483768","2"
"8","1","80","2","51.4044535002183","2"
"10","1","80","2","52.368660795627086","2"
```

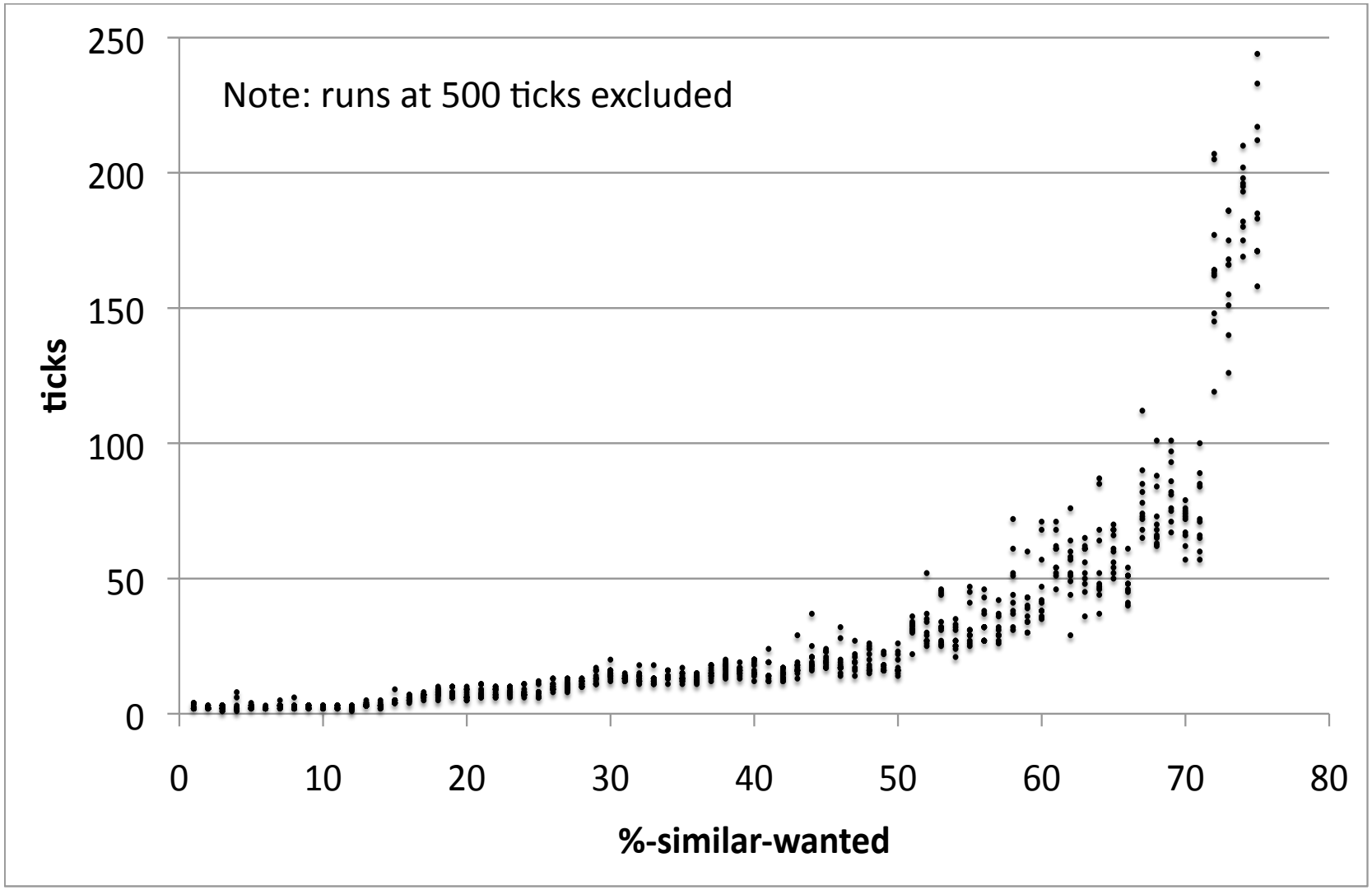
… continued with one line for each of the 1000 runs

You can load this into most statistical packages, such as R or Excel spreadsheet

Scatter plot (in excel)

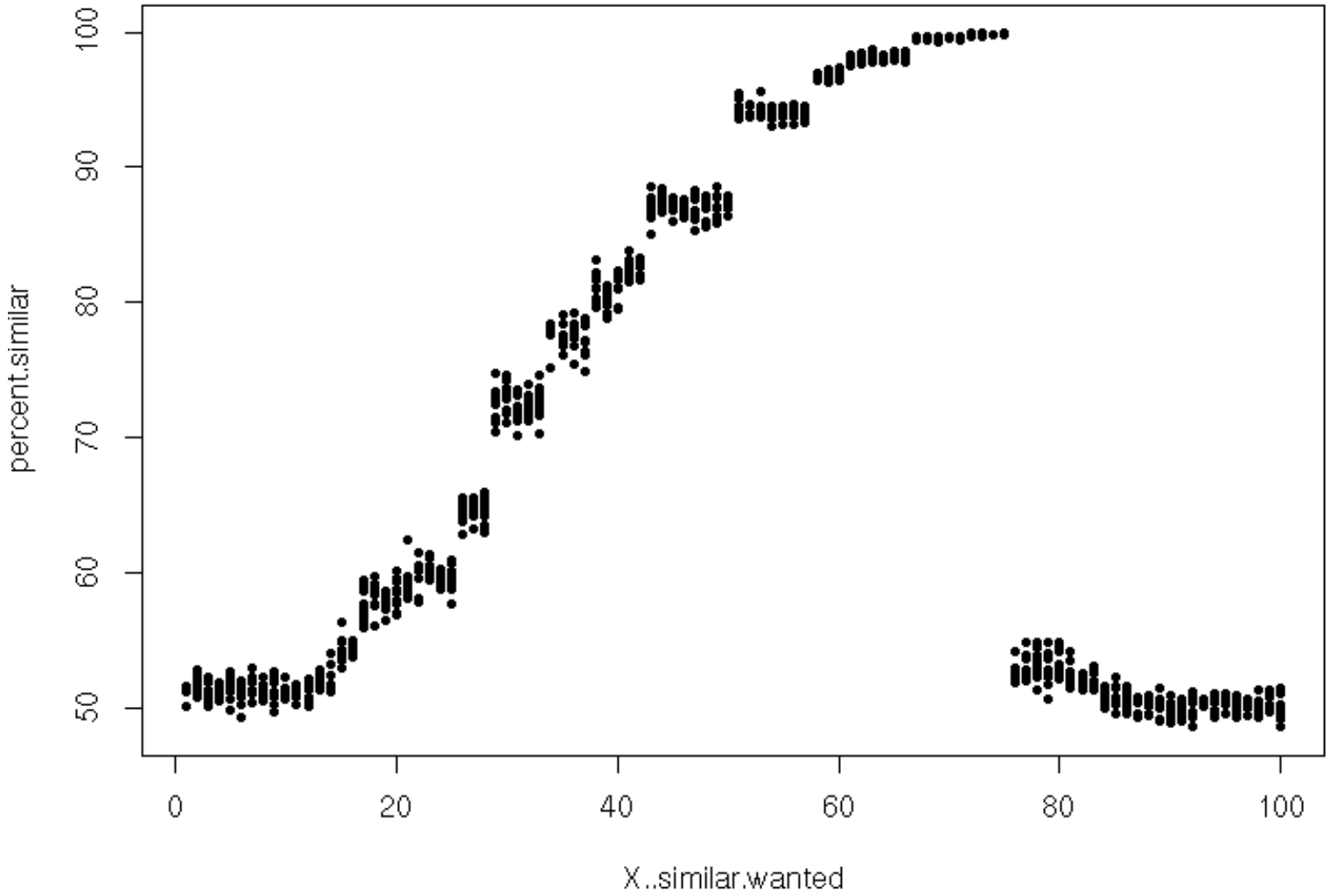


Scatter plot of ticks



To do a simple plot in R type:

```
tr = read.csv("seg experiment-table.csv", skip = 6)
attach(tr)
plot (X..similar.wanted, percent.similar, pch=20)
```



Results – what have we found?

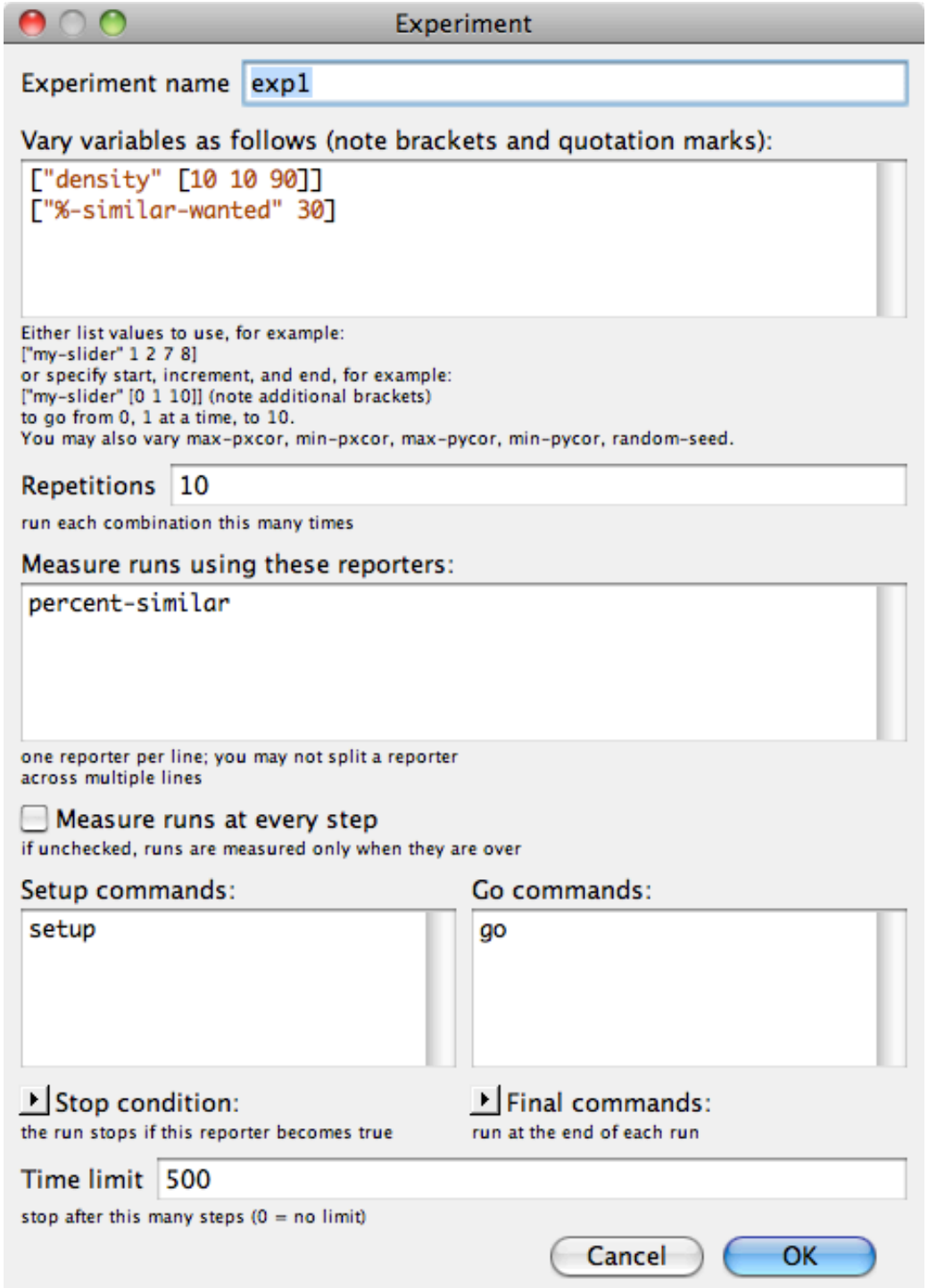
- Non-linear relationship between %similar-wanted (input parameter) and emergent segregation (the percent-similar output measure)
- With %similar-wanted > 75% no stability emerges which actually leads to lower segregation due to very low satisfaction levels and constant moving => random
- We could of course explore other parameters and other measures in this way

Task 1 – scan of density experiment

- Perform a sensitivity analysis of the density parameter in the segregation model using a BehaviorSpace experiment to:
  - Hold %similar-wanted to 30
  - Vary density from 10 to 90 in steps of 10
  - Perform 10 runs for each density value
  - Collecting percent-similar measure at end of each run
  - Terminate runs after 500 ticks
  - Load result into some package and do a scatter plot of density against percent-similar*

*Note: A simple way to do this is to load the CSV file into the google sheets spreadsheet app. If you e-mail the file to yourself in gmail you can then click on the file attachment and open it as a sheet and produce a plot

Task 1:
Experiment setup should look something like this



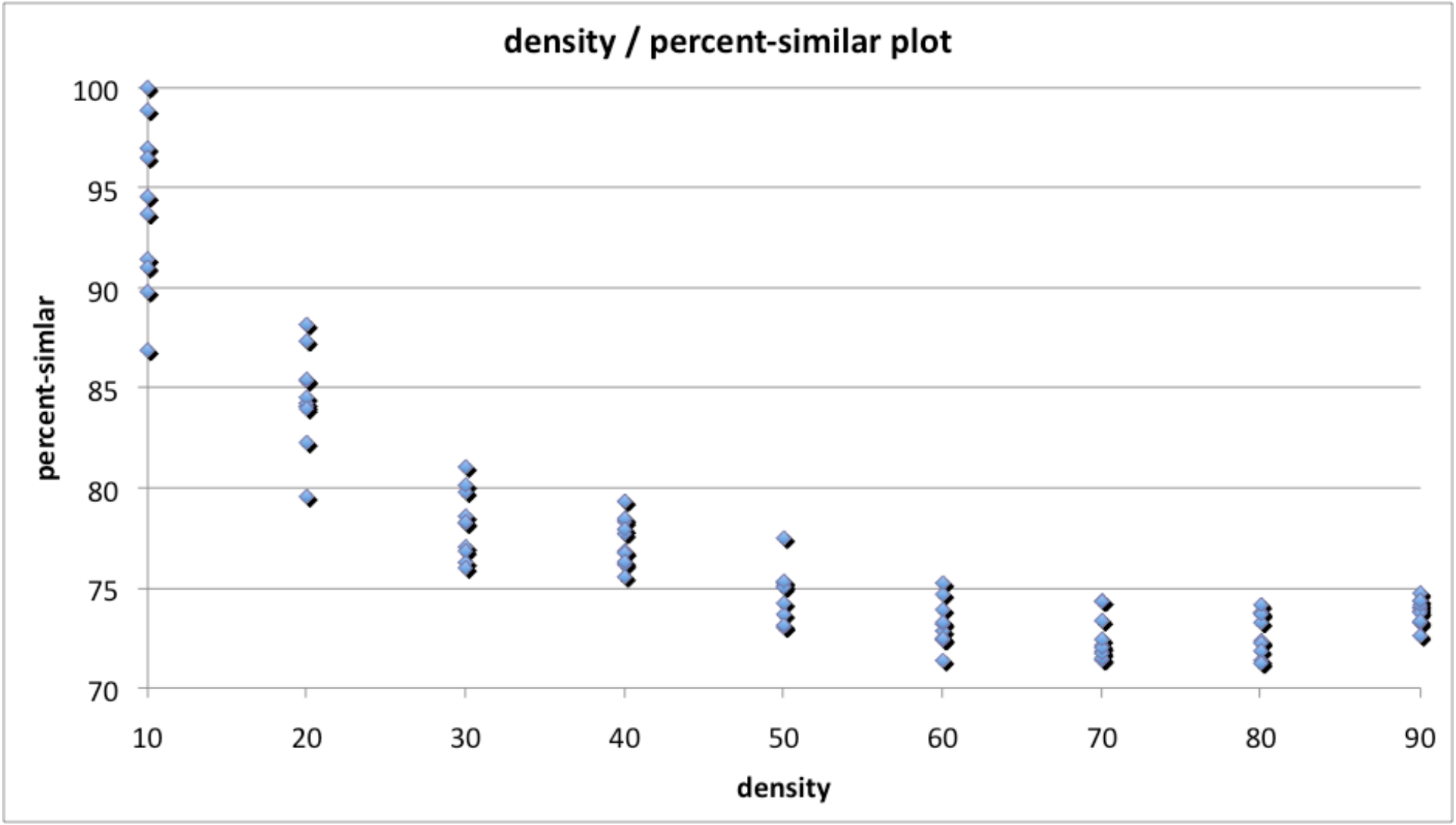
Task 1 – output csv file

Something similar to:

```
"BehaviorSpace results (NetLogo 5.2.0)"
"seg.nlogo"
"exp1"
"04/30/2016 21:07:28:790 +0200"
"min-pxcor","max-pxcor","min-pycor","max-pycor"
"-25","25","-25","25"
"[run number]","density","%-similar-wanted","[step]","percent-similar"
"1","10","30","4","93.10344827586206"
"2","10","30","8","96.875"
"3","10","30","7","91.42857142857143"
"4","10","30","7","88.76404494382022"
"5","10","30","5","98.30508474576271"
"6","10","30","7","91.30434782608695"
"7","10","30","4","95.04950495049505"
"8","10","30","6","93.16239316239316"
"9","10","30","7","95.83333333333334"
"10","10","30","5","90.9090909090909"
"11","20","30","9","85.57919621749409"
"12","20","30","9","82.67716535433071"
```

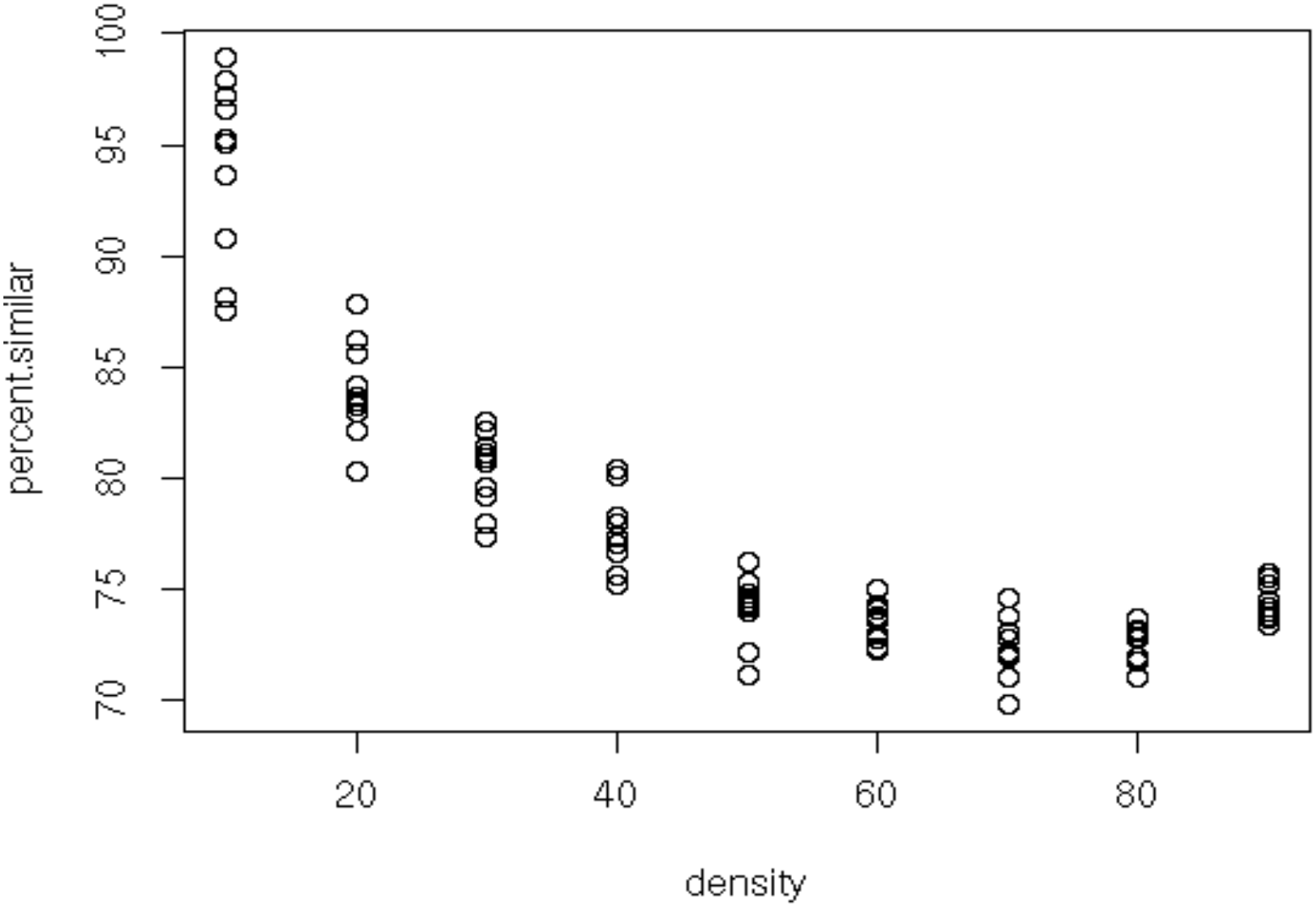
... continued for the 90 runs

Task 1 - scatter plot (excel):



To do a simple plot in R type:

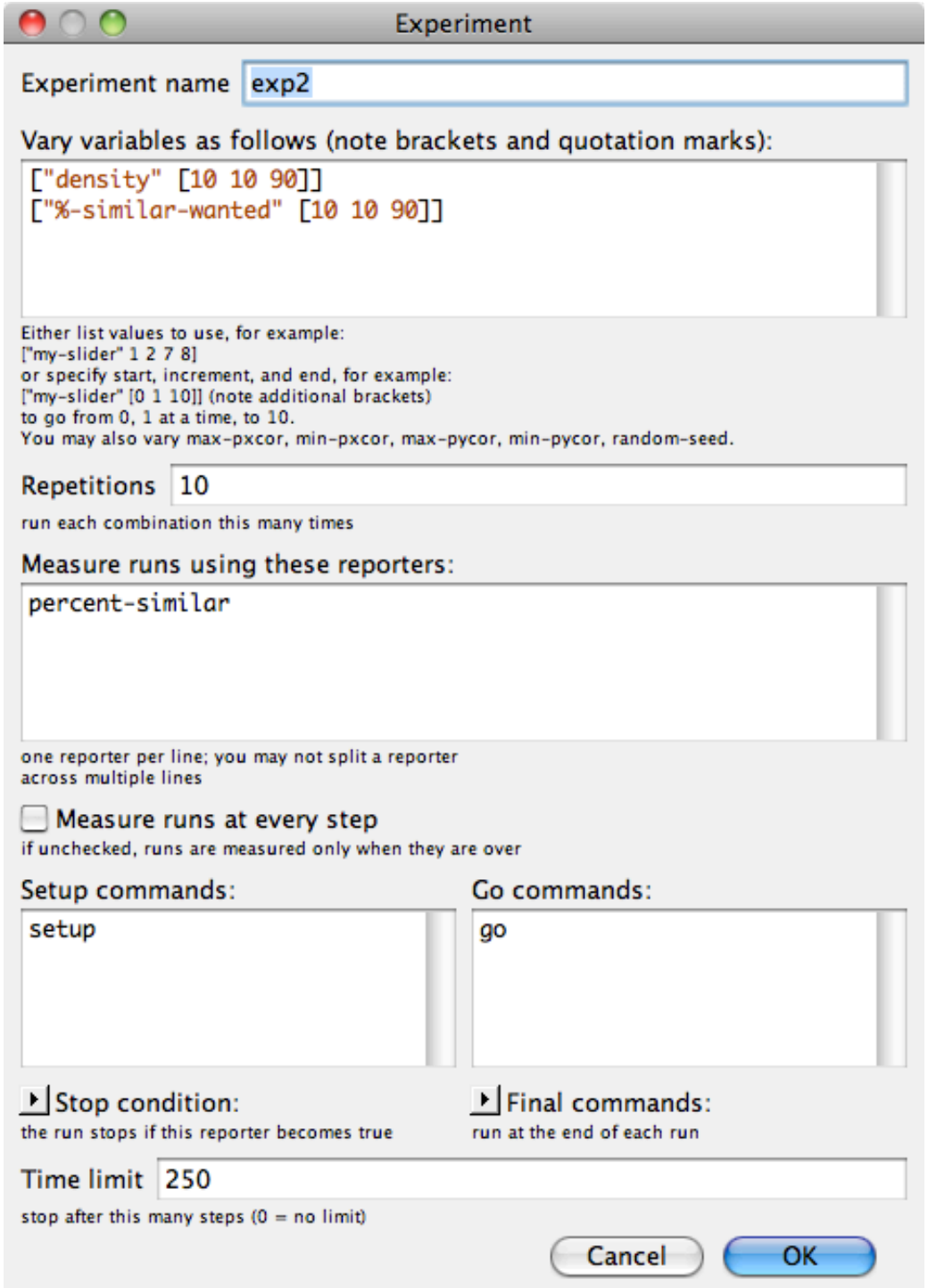
```
tr = read.csv("seg exp1-table.csv", skip = 6)
attach(tr)
plot(density, percent.similar)
```



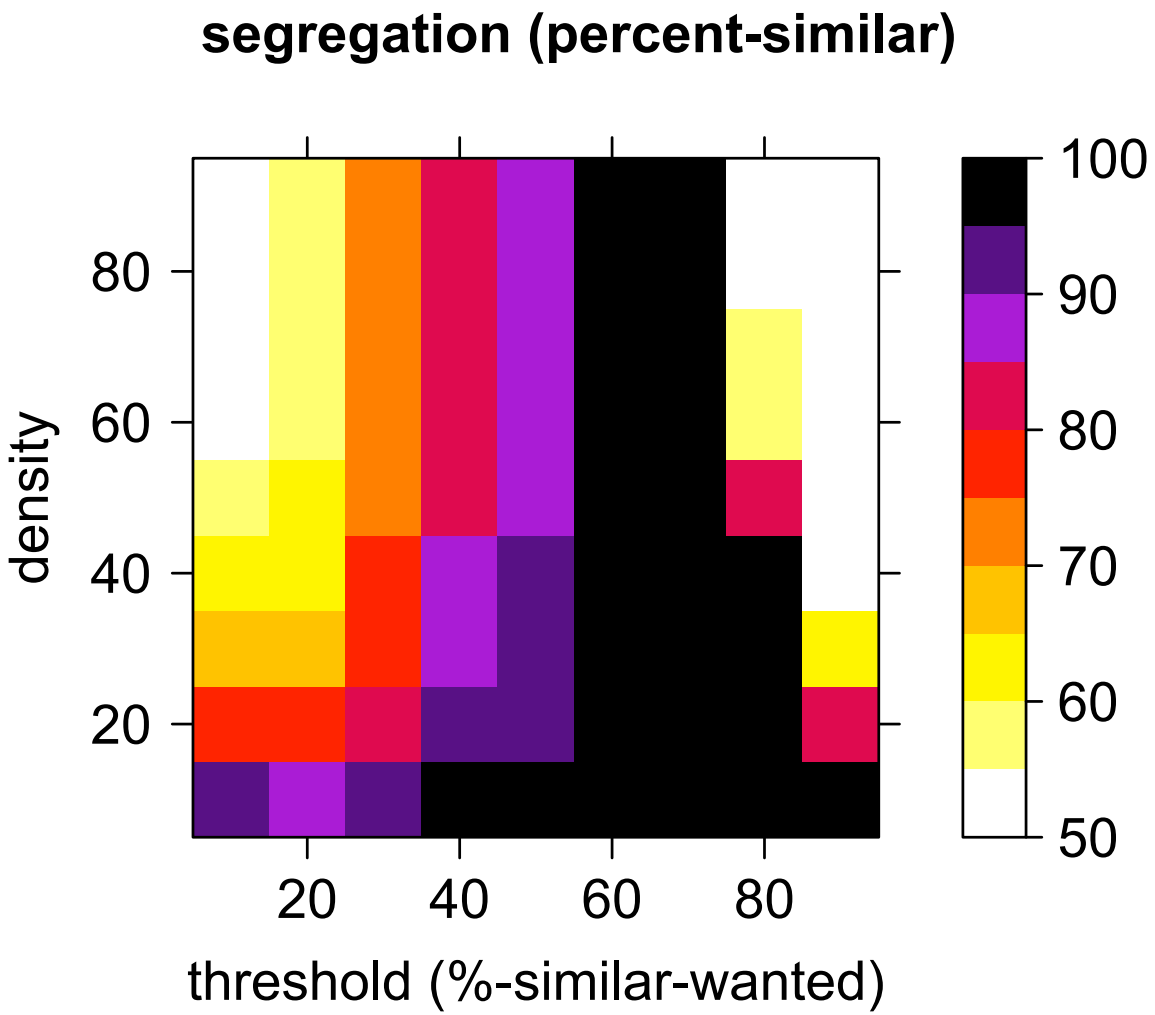
Vary several parameters

- We can vary several parameters at the same time to produce a scan of all combinations
- Suppose we wish to vary:
  - %similar-wanted between 10 and 90 in steps of 10
  - density between 10 and 90 in steps of 10
  - measure percent-similar for each combination
  - do 10 runs for each combination
  - terminate by step 250
  - giving 9 x 9 x 10 = 810 individual runs

Experiment setup for scanning density and %-similar-wanted together



Result – level plot of average percent-similar for different density and %-similar-wanted values



R script to produce level plot from csv file:

```
# read in netlogo csv file, skip 6 info lines at start
tr = read.csv("seg exp2-table.csv", skip = 6)
# select and rename fields (columns) and put into ts
# (R changes their names in csv file to remove illegal charaters)
ts <- data.frame(density = tr$density,
                 threshold = tr$X..similar.wanted,
                 segregation = tr$percent.similar)
# calaculate averages for each unique (density, threshold) pair
ta <- aggregate(ts, by=list(ts$density,ts$threshold), FUN=mean)
library(lattice)
p <- levelplot (segregation ~ threshold * density, data = ta)
print (p)  # display the plot
```

Note: the level plot on the previous screen uses a fancy script similar to this that improves the appearance.

Running “headless” experiments

- If you have a netlogo model in which you have defined an experiment
- You can run the experiment in headless mode as a background process (i.e. no display / user interface)
- This is useful if you wish to use a server machine with multiple cores
- You can leave the process running and come back to it later (perhaps days or weeks later!)

See: Netlogo Manual: Features > BehaviorSpace > Advanced usage

Headless experiments

- Given a machine with java installed
- Create a directory containing copy of:
  - NetLogo.jar file
  - lib directory (and all contents)
  - your-model.nlogo
- NetLogo.jar and the lib directory can be found in the NetLogo install directory
- your-model.nlogo is your model with an experiment defined in BehaviorSpace

Headless experiments

Within the directory you can type the following at the command line:

```
java -Xmx1024m -Dfile.encoding=UTF-8 -cp NetLogo.jar \
  org.nlogo.headless.Main \
  --model your-model.nlogo \
  --experiment your-experiment-name \
  --table result.csv
```

Where:

- *your-experiment-name* is the name you gave to the BehaviorSpace experiment in your-model.nlogo
- *result.csv* is the name of a file to write the output to

Headless experiments – shell script

If you intend to run headless experiments regularly then it helps to create a shell script that takes two arguments: <program-name> <experiment-name>

A shell script called nlr.sh:

```
#!/bin/bash
echo "Netlogo experiment \"$2\" in model \"${1}.nlogo\""
echo "output file: $1-exp-$2-table-out.csv"
nohup nice -n 19 \
java -Xmx1024m -Dfile.encoding=UTF-8 \
-cp NetLogo.jar org.nlogo.headless.Main \
--model ${1}.nlogo \
--experiment $2 \
--table $1-exp-$2-table-out.csv &
```

Then you can simply type:

```
./nlr.sh your-model your-experiment-name
```

Notice the use of “nohup” and “nice –n 19” (allows you to disconnect / reconnect and gives your jobs low priority so you don’t hog the server)

Integration with Java, R and Mathematica

- You can load CSV files into Java programs, R or Mathematica but there are tools for deeper, Java, R and Mathematica integration:
- Java controlling API:
  - https://github.com/NetLogo/NetLogo/wiki/Controlling-API
- R integration:
  - http://bergant.github.io/nlexperiment
- Mathematica integration:
  - Netlogo manual / features / mathematica link

Note: I have not used these so can not say how well they work. If anyone explores them I would be interested to know how well they worked

Other kinds of parameter exploration

- We only looked at simple scans of parameters
- If you want to something different such as:
  - Random sample of parameter space
  - Intelligent searching of parameter space (say using hill-climbing or other optimisation approach)
- Then you can:
  - write netlogo code to directly implement this (netlogo can write directly to files to store results for processing)
  - Or use the integration tools