Implementing Factorial Designs in Group Simulator

Jonathan H. Morgan

Fachhochschule Potsdam | University of Applied Sciences
IaF Urbane Zukunft | Institute for Urban Futures
Kiepenheuerallee 5, Haus 4, Raum 3.14
14469 Potsdam, Germany

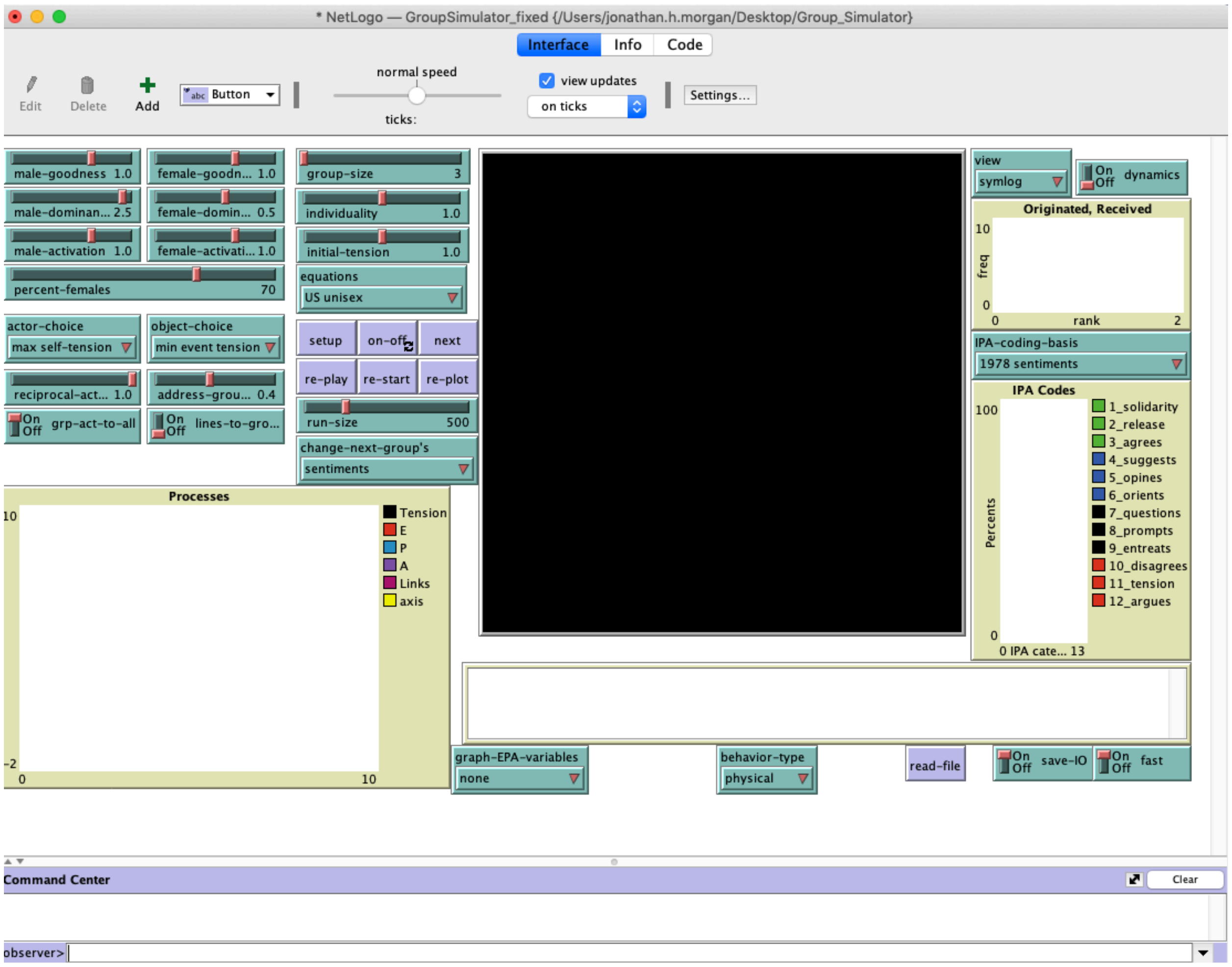
morgan@fh-potsdam.de

December 2, 2020

Summary: The purpose of this document is to supplement the Group Simulator Guide (Heise 2011). Although the guide provides a comprehensive description of Group Simulator's basic operation, features, and theoretical linkages between the simulation's design and classic studies of small-group behavior, it provides little guidance on how to efficiently run factorial designs in Group Simulator. Consequently, I focus here on the steps necessary to systematically vary simulation parameters, run Group Simulator using "headless" mode (i.e., without the GUI), and analyze Group Simulator's outputs.

Designing a Group Experiment in Group Simulator: Working with Behavior Spaces

One advantage of simulation is that it allows the researcher to explore systematically theoretical situations. For example, how does changing norms about addressing the group versus other group members influence the level of tension group members feel? We can compare these situations by varying the percentage of time on average group members address the group rather than each other. Exploring theoretical situations involves systematically varying the simulation's inputs. In Group Simulator's GUI mode, displayed in Figure 1, we do this by simply moving the slider in the case of continuous inputs like group size or selecting another radio button option in the case of categorical inputs such as which dictionary to use when computing the agent's optimal behavior each turn.



male-goodness 1.0
female-goodn... 1.0
group-size 3
male-dominan... 2.5
female-domin... 0.5
individuality 1.0
male-activation 1.0
female-activati... 1.0
initial-tension 1.0
percent-females 70
equations: US unisex
actor-choice: max self-tension
object-choice: min event tension
reciprocal-act... 1.0
address-grou... 0.4
run-size 500
change-next-group's: sentiments
view: symlog
IPA-coding-basis: 1978 sentiments
graph-EPA-variables: none
behavior-type: physical

Figure 1. Group Simulator User Interface.

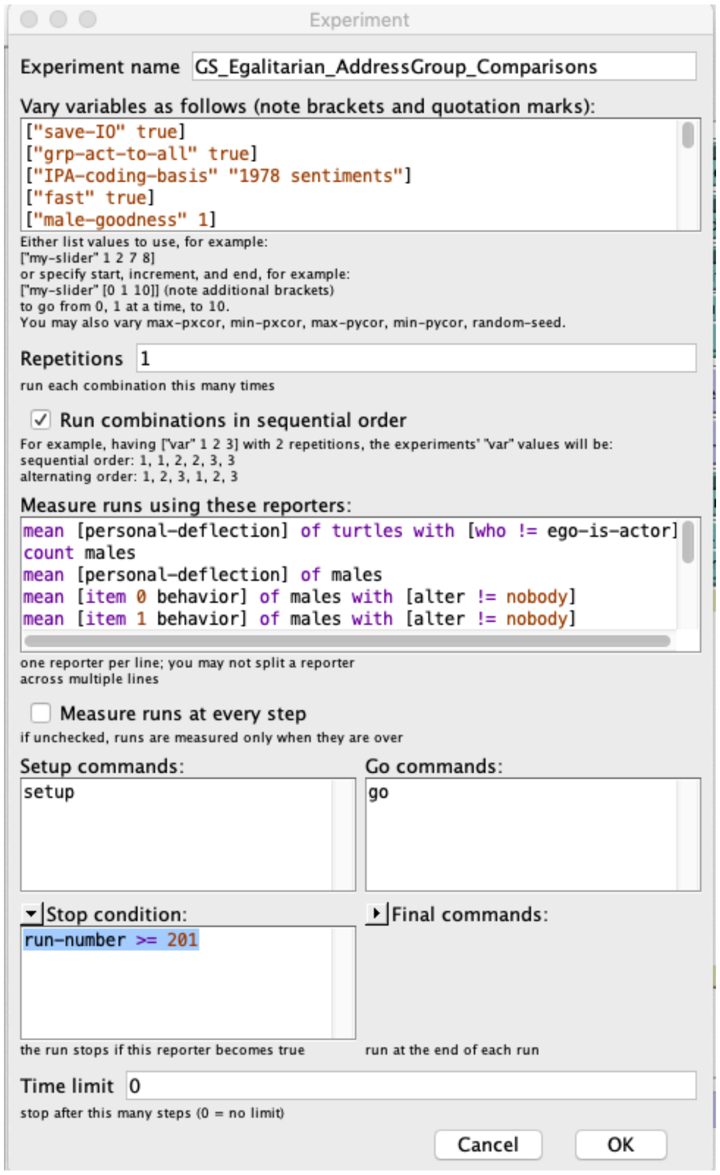
There are, however, multiple disadvantages with relying on the GUI to setup and run simulation experiments as the scale and complexity of the research question increases. First, when running the simulation using the GUI, you cannot directly control how many simulations will occur. You can control how many turns each simulation consists of with the *run-size* slider, but not how many simulations. Using the GUI alone, Group Simulator will produce as many simulations as there is time until the user stops the simulation by pushing the *on-off* button. In practice, this means that when running an experiment in Group Simulator using the GUI you have to monitor it, and that you are unlikely to replicate the exact number of simulations if you were to repeat the experiment.

Second, systematically varying more than a few settings quickly becomes impractical. Although you can speed-up the simulation by turning off the *dynamics* and using *fast* mode, you have to manually setup the simulation, monitor it, terminate it, change the variable of interest, and start the next simulation. You also have to carefully relabel the output file, otherwise Group Simulator will append the results to the existing file. The appending feature is, in itself, reasonable; but in many cases, the transition to a new condition is not clear in the data because a change in the setting changes tendencies not discrete values.

Fortunately, Netlogo provides a utility called the ***BehaviorSpace*** that enables the researcher to treat the simulation as a factorial experiment that runs until complete, significantly reducing the amount of handholding required to run the experiment. For more information regarding the how to use the BehaviorSpace utility, you can refer to the Netlogo manual: (https://ccl.northwestern.edu/netlogo/docs/behaviorspace.html).

We will focus here on how to use the BehaviorSpace utility to run experiments in Group Simulator. To access the utility, select ***Tools*** from the Netlogo menu bar. Next, select ***BehaviorSpace***. At this point, you will have the option to create a ***New*** experiment, ***Edit*** an experiment, ***Duplicate*** an experiment, ***Delete*** an experiment, or ***Run*** an experiment.

To illustrate how the utility works, we are going to choose Edit, selecting the GS_Egalitarian_AddressGroup_Comparisons. When you choose this experiment, you should see Figure 2, shown below.


Experiment

Experiment name GS_Egalitarian_AddressGroup_Comparisons

Vary variables as follows (note brackets and quotation marks):

```
["save-IO" true]
["grp-act-to-all" true]
["IPA-coding-basis" "1978 sentiments"]
["fast" true]
["male-goodness" 1]
```

Either list values to use, for example:
["my-slider" 1 2 7 8]
or specify start, increment, and end, for example:
["my-slider" [0 1 10]] (note additional brackets)
to go from 0, 1 at a time, to 10.
You may also vary max-pxcor, min-pxcor, max-pycor, min-pycor, random-seed.

Repetitions 1
run each combination this many times

[✓] Run combinations in sequential order
For example, having ["var" 1 2 3] with 2 repetitions, the experiments' "var" values will be:
sequential order: 1, 1, 2, 2, 3, 3
alternating order: 1, 2, 3, 1, 2, 3

Measure runs using these reporters:

```
mean [personal-deflection] of turtles with [who != ego-is-actor]
count males
mean [personal-deflection] of males
mean [item 0 behavior] of males with [alter != nobody]
mean [item 1 behavior] of males with [alter != nobody]
```

one reporter per line; you may not split a reporter across multiple lines

[ ] Measure runs at every step
if unchecked, runs are measured only when they are over

Setup commands:

```
setup
```

Go commands:

```
go
```

Stop condition:

```
run-number >= 201
```

the run stops if this reporter becomes true

Final commands:

run at the end of each run

Time limit 0
stop after this many steps (0 = no limit)

Figure 2. BehaviorSpace Experiment Window.

We will walk through each experimental element from top to bottom. The ***Experiment name*** differentiates one experiment from another; it is a required because a Netlogo simulation can support multiple experiments. For example, GroupSimulator_Fixed.netlogo consists of 5 experiments, four experiments examining interaction norms and a diagnostic experiment implemented by David Heise when testing Group Simulator's various features. Note, the diagnostic experiment takes days to run.

The ***Vary variables as follows*** window specifies what settings the simulation will treat as constants and which the simulation will vary and in what order. If the setting is left unspecified, then Group Simulator will provide a default. I, however, recommend specifying each parameter displayed in the GUI to be absolutely certain of your simulation's settings, and to facilitate writing-up it up or responding to questions about the simulation in the future—specifying the inputs provides a written artifact you can reference in the future.

```
["save-IO" true]

["grp-act-to-all" true]

["IPA-coding-basis" "1978 sentiments"]

["fast" true]

["male-goodness" 1]

["male-dominance" 1.5]

["male-activation" 1]

["female-goodness" 1]

["female-dominance" 1.5]

["female-activation" 1]

["percent-females" 0]

["group-size" 3]

["individuality" 1]

["initial-tension" 1]

["actor-choice" "max self-tension"]

["object-choice" "min event tension"]

["address-group-Pr" 0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1]

["reciprocal-act-Pr" 0.7]

["equations" "US unisex"]

["run-size" 500]

["change-next-group's" "sentiments"]
```

Each setting has its own line. Specifying a setting consists of indicating what parameter is being specified and at what value. The parameter comes first and is in quotes. All characters following the quotes are considered values for the specified input. Values take three major forms: as numbers in the case of continuous inputs such as group-size, as Boolean values true or false such as whether the simulation should save the input-output log, and as quoted text if the value is a category such as what dictionary to use when making calculations. The parameters that are varied in the experiment take multiple values. For example, we vary the address-the-group percentage by 0.1 increments from 0 to 1 which corresponds to the following row of values: 0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1. For parameters that are treated as constants, only one value appears such as the initial tension agents experience, 1 in this case. Finally, brackets encapsulate both the parameter name and its values, signaling to Group Simulator to treat this block of characters as a unit.

The **Repetitions** window allows the analyst to have Netlogo run the model more than once at each combination of settings by entering a number greater than one. Although seemingly intuitive, the Repetitions command can generate non-intuitive output files. If, for example, the objective is to run factorial blocks of 200 simulations consisting of 500 turns for each address-the-group percentage, there are two approaches. The first approach is to set ***Repetitions*** to 200 and a stop condition where ***run-number*** >= 1. The problem with this approach is that the resulting output files can mix rows associated with different combinations because Group Simulator's write functions are not designed to accommodate parallel processing. Instead, I recommend a Repetition of 1 and setting Group Simulator to stop once the run-number is equal to or greater than 201. The advantage of this approach is that each combination is treated as its own block and written-out sequentially. Note, select ***Run combinations sequential order*** to ensure that this is the case. Finally, to make interpreting the log files easier, I recommend conducting simpler experiments. The effort of evoking one versus two or three experiments is not that great; and, the potential for IO headaches is much less.

In addition to Group Simulator's run and action files, the BehaviorSpace utility can generate a customizable summary file. In practice, I have found that Group Simulator's run and action files record the information I need. But, the BehaviorSpace files do have two advantages. One, you can have parameters that normally act behind the scenes output values to this file. Two, the analyst can specify that the file be outputted as a CSV whereas Group Simulator's default files require formatting to analyze. I have found that these files can be useful for checking that experiment is operating as intended.

To specify what aggregate measures the BehaviorSpace utility will report, you use the ***Measure runs using these reporters*** window. Netlogo is based on Java, an object-oriented program. Reporters report the state of objects: turtles (agents), patches, observers, or the global state. In the first line, I ask agents who are not the actor to report their personal deflection. In the other commands, I ask Netlogo to report the behaviors, current-transients, and emotions of all objects who are male agents because this simulation consists of only one type of agent. For the hierarchical experiments where there are leaders and followers, I specify duplicate commands for female agents.[1] The item numbers correspond to the number of values associated with that feature. For example, behaviors consist of three items, one for each EPA dimension. Because Java is a zero-indexed language, the first item is 0 rather than 1. Finally, if you want summaries outputted at the end of each run rather than at the completion of all runs, you can select the

[1] Male and female are placeholders for any two EPA profiles. For example, we use female to represent the identity boss, and male for the identity client in the hierarchical simulation experiments included in GroupSimulator_fixed.netlogo.

Measure runs at every step radio button. I typically do not do this because it significantly slows down processing.

```
mean [personal-deflection] of turtles with [who != ego-is-actor]

count males

mean [personal-deflection] of males

mean [item 0 behavior] of males with [alter != nobody]

mean [item 1 behavior] of males with [alter != nobody]

mean [item 2 behavior] of males with [alter != nobody]

mean [item 0 current-transients] of males with [alter != nobody]

mean [item 1 current-transients] of males with [alter != nobody]

mean [item 2 current-transients] of males with [alter != nobody]

mean [item 0 emotion] of males with [alter != nobody]

mean [item 1 emotion] of males with [alter != nobody]

mean [item 2 emotion] of males with [alter != nobody]
```

Where do I find the object names and features necessary to generate a report? The answer is by selecting the ***Code*** radio button in Group Simulator Interface. By selecting this button, the analyst can examine the underlying code directly. At the top of the script, there is a list of global objects and features and a list of agent-level features. If you do not find what you need here, you can search the script using key words such as transients. Unless you are modifying Group Simulator, you are unlikely to need to use this functionality. Nevertheless, it can come in handy such as when you implement a new set of EPA equations.

Netlogo requirs separate commands for setting-up and starting the simulation, the ***Setup command*** and ***Go commands*** respectively. The first command sets initial values; the second begins the first turn of the simulation. In Group Simulator, these commands are ***setup*** and ***go***, nothing fancy here. These commands will be the same for all experiments.

The ***Stop condition*** specifies what conditions must be met to stop the simulation. Depending on the simulation, these can vary in complexity. In the case of Group Simulator, I apply a simple condition involving the run-number counter. Run the simulation until the run-number counter is greater than or equal to 201. Combined with the ***Vary variables as follows*** commands, Group Simulator will run 11 (one for each tenth percentile) x 500 (turns) x 200 simulations, resulting in 1,100,000 iterations.

Finally, the ***Time limit*** window specifies when to end the craziness. The ***Time limit*** command can be useful as a sanity check. If a change to the simulation is resulting in slower performance, this command can help to benchmark just how much slower. When running experiments as opposed to troubleshooting a new feature, I generally do not use this option.

Running Experiments "Headless"

Netlogo's supports calling Netlogos remotely from another application and executing a specified simulation. Netlogo refers to this mode as, "Headless" mode (see https://ccl.northwestern.edu/netlogo/docs/behaviorspace.html for more information).

Headless mode is useful for looping through multiple simulation experiments. The advantage of this approach is that you can specify simple experiments (reducing potential IO headaches) while also executing the full factorial design with minimal handholding.

I have found that Python's *subprocess* functionality to quite useful when working with Headless mode. Consequently, I work through an example of Headless mode using a Python script, but you can easily call Python from another language such as R or Julia which is my usual workflow.

To use Headless mode, you first have to specify the following locations: 1) where your copy of Netlogo is located, 2) where the simulation file you intend to run is located, and 3) where to output the aggregate data file specified in the BehaviorSpace.

- home_netlogo = "/Applications/NetLogo 6.1.0/netlogo-headless.sh"
- home_path = "/Users/jonathan.h.morgan/Desktop/Group_Simulator/GroupSimulator_fixed.nlogo"
- output_csv = "/Users/jonathan.h.morgan/Desktop/Group_Simulator/output.csv"

Next, you specify the experiments you wish to execute. In this case, I am specifying four experiments in a character array that I intend to run sequentially.

```
experiments_array = c("GS_Egalitarian_AddressGroup_Comparisons",
"GS_Egalitarian_Reciprocity_Comparisons", "GS_Hierarchical_AddressGroup_Comparisons",
"GS_Hierarchical_Reciprocity_Comparisons")
```

Finally, we are ready to call Netlogo from Python. In the example, I am executing Python script from another application which is why it's encapsulated by py""" """.

```
py"""


import subprocess

subprocess.run([$home_netlogo, "--model",

$home_path,

"--experiment", $experiment,

"--threads", "1",

"--table", $output_csv])


"""
```

In the first line, I am importing the subprocess package into my local instance of Python. I am then using the subprocess.run command to call Netlogo. The dollar signs indicate these are parameters that I have specified in R prior to calling Python. In the second line I am specifying where to find the simulation script I wish to execute. The commands listed in the following lines are keywords, "—model", "—experiment", "—threads", and "—table", that correspond to model or simulation file, the simulation experiment I intend to run, the number of threads I intend to use, and the name of the output CSV generated. The output csv corresponds to the CSV output specified in the BehaviorSpace not the text file that Group Simulator generates automatically. Keep this in mind when writing your loop.

Formatting and Analyzing Group Simulator Files

Finally, we come to the simulation output. Group Simulator output files are very logical, but not very user friendly. Group Simulator outputs two files by default: *data_GroupSimulator_actions.txt* and *data_GroupSimulator_runs.txt*. Over the years, I have written multiple scripts in multiple languages to parse both these file types. I am going to focus on a relatively straightforward R script, *Rcode for formatting GS_5Dec2020.R*, written almost entirely in Base R to avoid versioning issues, although there are no guarantees with R. I use two of R's most widely used packages, ggplot and dplyr, to generate some example visualizations at the end of the script, but these examples are not necessary to perform the script's central formatting function.

Rcode for formatting GS_5Dec2020.R formats Group Simulator's action files. I focus on action files because all the information in the Group Simulator's run files can be aggregated from its accompanying action file. The script supports parsing multiple actions files. If an action file includes multiple simulation experiments (indicated by the fact that there are multiple parameter lines in the file), the script parses the file and distinguishes the data associated with each experiment with the *sim_id* variable. The script generates two major outputs: a simulations list and a parameters list. The simulations list includes the data for each parsed action file as a data-frame, with each data-frame being a list element. The parameters list includes the settings and variables for each simulation, again each list element corresponds to a parsed action file.

The analyst has the option to save both lists as R objects. In list form, it is easy to perform iterative operations either using R's lappy functionality or using loops. If, on the other hand, you are collaborating with someone using different software to analyze the data. You can simply output the list element as CSVs in the case of the simulation data and as a text file in the case of the parameters data. I provide an example outputting each file type in *Rcode for formatting GS_5Dec2020.R*.