On the Open-Endendness of Detecting Open-Endedness : Supplementary documentation of code and data

Susan Stepney[1,2] and Simon Hickinbotham[1,3]

[1]York Cross-disciplinary Centre for Systems Analysis
[2]Department of Computer Science, University of York, UK
[3]Department of Electronic Engineering, University of York, UK

26 July 2022

1 Introduction

This document provides the location of data and code used to produce the paper: Stepney and Hickinbotham, "On the Open-Endendness of Detecting Open-Endedness", *Artificial Life*, 2023. It outlines the code functions, and documents the data file formats.

The paper uses a dataset previously generated for Hickinbotham et al., 2021, which is referred to here as the *input data*. This dataset is analysed to create a new dataset for the current paper, and is referred to here as the *output data*.

2 Input data

The dataset used as input data in this paper is available from https://doi.org/10.15124/305dfdb6-9483-4c5b-8a01-c030570b9c31. The data is in two forms:

- Logfiles from the original stringmol experimental runs
- Datafiles generated from logfiles, created for Hickinbotham et al., 2021

Note the latter are generated from the former: this is a lengthy process and it is not recommended to repeat it.

3 Output data

The process of generating the output data from the input data (§2) takes over 10 hours on a modern laptop. The output data generated as the basis of the

analysis in this paper is available from
https://doi.org/10.15124/88a8bad0-b23f-4afc-80a6-77e6358b7a8f.

4 Code producing the output data

The R package 'RStringmol' was developed and used to generate the output data files (§3) used in this paper. This package is available from https://github.com/uoy-research/Rstringmol. That page has instructions for installing the package, including how to build tutorial vignettes. The R vignette "stringmol analysis II" describes how to generate the output data files from the input data. See the code repository README for instructions on viewing this vignette.

5 Code analysing the output data files

This section describes the code developed to analyse the output data files and produce the tables and plots in the paper. The results in the paper were produced using python v3.5.2, matplotlib v3.0.3, numpy v1.18.5, and pandas v0.24.2. This python code is available in the python subdirectory of the Rstringmol package described in §4.

There are three python files used to produce the tables and plots in the paper. There are two utility files: `oee_utils.py`, functions for finding and reading data files; `oee_plot.py`, functions for plotting. The main file `figures.py` imports these, and produces all the plots and tables, from the data files, using the following functions:

- `qnn_plot(start, end)`, QNN for runs `start–end`. Prints total QNN values, plots of QNN (paper figure 2), and plots of cumulative QNN (figure 3)
- `pop_size()`, population size for runs 1–8. Prints minimum and maximum population size per run (figure 5), plots of population size (figure 4.top) and boxplots of ranges of populations size (figure 4.bottom)
- `species_count()`, species counts for runs 1–8. Prints minimum and maximum species counts in first and second half of runs (figure 7, min and max columns), plots of total and new species counts (figure 6.top) and boxplots of ranges of species counts (figure 6.bottom)
- `species_12abundance()`, species abundances for runs 1–8. Prints number of unique species per run (figure 7, unique column), stacked plots of individuals per species (figure 8)[1], plots of 12 most abundant species (figure 9), prints data on 12 most abundant species (figures 18–25)
- `length_lifetime(run)`, scatter plot of string length against lifespan, for a given run (figure 10)
- `string_length_diversity(run)`, density plot of string length against time, for a given run (figure 11)

[1] The code produces large pdf files, as there are 30–60 000 lines per plot. The plots in the paper are jpeg files produced from screenshots of the pdf files.

- `self_repl(run)`, stacked plot of self-replicator and other reaction types against time, for a given run (figure 12)
- `repl_reactions(run)`, stacked plot (figure 13) and log plot (figure 14) of parasitic and hypercycle reaction types against time, for a given run
- `toggle_opcodes(run)`, density plot of toggle opcode use against time, for a given run (figure 15)

6 Experimental run data file formats

The output data files (§3) from which the plots are produced are stored in subdirectories of the top level directory `rundata`. This top level directory should be placed in the same directory as the python code `figures.py`.

Each subdirectory `rundata/runs/run<N>`, where `N` is the run number, `1`–`8`, contains all the datafiles for that run.

The output data from each run comprises a set of files, one for each logged timestep. One logfile is recorded each 20 000 timesteps. For an experiment that runs to completion, the final timestep is 2 000 000, so there are 100 or 101 (if the initial state at time zero is logged) files.

There are two kinds of output data files in the subdirectories: species data files and reaction data files. The files are csv files. Each row is the data for a species or reaction, as specified. The columns provide a variety of measures for each row. For each column, we give the column header (as appearing in the csv file), the type of the measure, and a comment on the meaning of the measure.

6.1 Species data files

In species data files, the focus is on the *strings* present at a given timestep, and on the properties of those strings. There are three kinds of species data files, described in detail below.

- `species<0N>.csv` is a list of species. `<0N>` is the run number, `01`–`08`.
- `sppcounts<TTTTTTT>.csv` is details of the species, one per logged timestep. `<TTTTTTT>` is the timestep number, `0000000`–`2000000`.
- `popdy<0N>.csv` is a summary of the data above. `<0N>` is the run number, `01`–`08`.

6.1.1 List of species

The file `species<0N>.csv` is a list of the species observed in run `<N>`. Each row is a species; columns are:

1. `ID` : Str : species ID, unique within a run, `sp<N>`, where `<N>` =1..number of observed species
2. `start` : Int : the first timestep this species is logged (multiple of 20000)
3. `end` : Int : the last timestep this species is logged (multiple of 20000)
4. `n` : Int : total number of instances of the species observed, as 'string1', 'string2', or 'unbound' (as recorded in the logfiles; many more instances

will have appeared and then disappeared between logged timesteps, and so not been recorded)
5. `seq` : Str : the actual sequence of the species

6.1.2 Species data

Each file `sppcounts<TTTTTTT>.csv` lists species data for the species observed at that logged timestep. Each row is a species; columns are:

1. `t` : Int : the logged timestep (also recorded in the filename); included here to allow files to be concatenated
2. `ID` : Str : species ID as described in §6.1.1, column 1
3. `n` : Int : total number of instances of the species observed in this logged timestep
4. `unb` : Int : number of instances unbound (not reacting)
5. `act` : Int : number of instances that are 'string1' in reactions
6. `pas` : Int : number of instances that are 'string2' in reactions
7. `seq` : Str : the sequence, as given in the list of species (§6.1.1, column 5); repeated here as useful when examining the data by eye.

6.1.3 Summary species data

The file `popdy<ON>.csv` is a summary of the species data files, used for calculating evolutionary activity. Each row is a species at a logged timestep; (unlabelled) columns are:

- Int : the logged timestep
- Str : species ID (see §6.1.1, column 1)
- Int : Int : total number of instances of the species observed at the logged timestep, as 'string1', 'string2', or 'unbound'

6.2 Reaction data files

In reaction data files, the focus is on the *reactions* occurring at a given timestep, and on the properties of the specific reactions.

See Stepney and Hickinbotham (2021) for formal definitions of reaction and network properties mentioned here.

6.2.1 Observed reactions

The file `oeeRun<N>props<TTTTTTT>.csv` is a list of reactions logged in run `N`, 1–8, at logged timestep `TTTTTTT`, `0020000`–`2000000`. Each row is a reaction; columns are:

1. `actseq` : Str : sequence of string1
2. `passeq` : Str : sequence of string2
3. `product` : Str : sequence of the first product sting, or `empty` (other products are not recorded)
4. `pp_ActiveMod` : Bool : string1 is modified during the reaction
5. `pp_PassiveMod` : Bool : string2 is modified during the reaction

6. pp_SelfMod : Bool : pp_ActiveMod or pp_PassiveMod
7. pp_NoProduct : Bool : true if no product string is produced, (product == empty)
8. pp_NewProduct : Bool : true if product string is distinct from string1 and string2, (not(product == actseq or product == passeq))
9. pp_biolRep : Bool : true if replication reaction, (pp_Repl1 or pp_Repl2)
10. pp_SelfReplicator : Bool : true if self-replication reaction
11. pp_Repl1 : Bool : true if string1 is replicated
12. pp_Repl2 : Bool : true if string2 is replicated
13. pp_Jumper : Bool : true if jumper reaction
14. actlen : Nat : length of string1, #actseq
15. paslen : Nat : length of string2, #passeq
16. outputA : Str : sequence of string1 at end of reaction (different from actseq iff pp_ActiveMod)
17. outputP : Str : sequence of string2 at end of reaction (different from passeq iff pp_PassiveMod)
18. ccopy : Nat : count of number of executions of the copy operator
19. cmove : Nat : count of number of executions of the move operator
20. cover : Nat : count of number of executions of the copy operator where an opcode is overwritten, as opposed to appended to the end, (cover ≤ ccopy)
21. ctogg : Nat : count of number of executions of the toggle operator
22. bprob : Float : bind probability
23. nsteps : Nat : number of steps in the (non-mutating *in vitro*) reaction
24. nprod : Nat : number of product molecules, rarely more than 1
25. oneway : Bool : true if assignment of string1 is fully deterministic, dependson bind position (same as dbind)
26. dexec : Bool : true if execution is fully deterministic, in that there are no 'soft goto' cases
27. dbind : Bool : true if assignment of string1 is fully deterministic, which depends on bind position (same as oneway)
28. nobs : Nat : number of observations of this reaction in this timestep
29. np_Parasite2 : Bool : true if string1 (replicator) replicates string2 (parasite), but *vice versa vice versa*
30. np_Parasite1 : Bool : true if string2 (replicator) replicates string1 (parasite), but not vv
31. np_Parasite : Bool : true if a parasitic reaction, (np_Parasite2 or np_Parasite1)
32. np_MutualRepl : Bool : true if reaction is (half of) a mutual replication network
33. np_Hypercycle : Bool : true if reaction is (part of) a hypercycle network

6.2.2 Self-self reactions

In self-self reaction data files, the focus is on the species observed throughout the run, and on the properties of the self-self reactions of these species. These self-

self reactions may never have been observed during a run, but their properties are needed for some analyses.

The file `selfself0N.csv` where `N` is the run number, `1–8`, comprises self-self reaction data from the species observed in an entire run. Each row is a species, with data from self-self reactions of that species.

The columns are as in §6.2.1 (up to column 28, `nobs`). For each species, both string1 and string2 are individuals of that species: (`actseq` == `passeq`).

6.2.3 Evolutionary activity data files

The file `qnn<0N>.csv` gives the calculated QNN for each species for each logged time point. Each row is a species at a timestep; (unlabelled) columns are:

1. Int : the logged timestep
2. Str : species ID
3. Float : the QNN value for that species at that execution timestep

References

Hickinbotham, Simon, Susan Stepney and Paulien Hogeweg (2021). Nothing in evolution makes sense except in the light of parasitism: evolution of complex replication strategies. *Royal Society Open Science* **8**(8):210441. DOI: 10.1098/rsos.210441 (see p. 1).

Stepney, Susan and Simon Hickinbotham (2021). What is a Parasite? Defining reaction and network properties in an open ended automata chemistry. *ALife 2021, Prague, Czech Republic (virtual), July 2021*. MIT Press, pp. 598–606. DOI: 10.1162/isal_a_00413 (see p. 4).