PSM Fall 2019, Exercise Sheet 1 – (partially) solved

Problem 1. In insect societies the ratio of female vs. male individuals varies dramatically between species. A myrmecologist[1] wants to determine the f/m ratio for the ant species *Cataglyphis bicolor* (one of the most intensely studied ants – being one of the most heat-tolerant multicellular animals known and also being able of stunning navigation feats https://en.wikipedia.org/wiki/Sahara_Desert_ant). Your task: Describe in plain but precise English a suitable RSOI and its OO's; think of RVs that the myrmecologist will want to use; and formal specify the DVS for each of your RVs. *Note*: there are several natural ways to specify a RSOI for this scientific situation.

Solution (sketchy). Similar to the Speed-of-Light I example from the lecture notes, one option for a reasonably RSOI is to define it around this particular myrmecologist (call him *A*) and a particular field trip to the Sahara. The RSOI would then be quite narrowly described as something like the collection of *Cataglyphis* societies living in the area of the Sahara visited by *A* in the year of his travel. A more comprehensive RSOI would focus on generality and replicability and be defined without reference to a particular expedition as the collection of all *Cataglyphis* colonies in the past and future.–- A sensible data value space would be the set of nonnegative rationals (for f/m ratio values obtained from counts). This can be refined in various ways, for instance respecting that the f/m ratio in a colony may vary over the colony's lifetime, then one would also include the colony's lifetime status into the DVS, for instance adding a new RV with values in {colony_freshly_founded, colony_at_peak_development, colony_in_final shrinking_stage}.

Problem 2. Another exercise in setting up a RSOI and its components. – You know that the "random" number generating function `rand`, which comes with every programming language (including MS Word!) in one form or the other, is actually not really generating *random* numbers, but only *pseudorandom* numbers which look and feel like being drawn from the uniform distribution on [0 1], but which in fact are computed by a deterministic algorithm. Many such pseudorandom generating algorithms are known, and they differ with regards to aspects like speed, breakability, and importantly, they differ with respect to how close they come to true randomness. Put yourself into the role of a software engineer who is about to program some simulation toolbox for some end-customer, and this toolbox must contain a high-quality `rand` function call. You have the choice between ten different pseudorandom number generating algorithms A_1, …, A_{10} that you have found in the mathematics literature. You want to do some practical test-checking for finding out how these algorithms compare to each other. Your task: specify an appropriate RSOI-OO-OP-OA-DVS scenario.

Solution: I don't give further hints here – you are on your own. For a discussion of solutions come to the tutorial!

[1] Myrmecology: the science of ants