Evolution of communication systems

BSc Psychobiology, UvA

► These are actions for you to do

? These are questions

⋆ This is a question that could be on the exam

1 Goals

In today's computer lab you will experiment with simulated evolution and look at a simple model of the evolution of communication. The goals are to

- Better understand the concepts of genotype, genotype space, fitness, fitness landscape, selection, mutation, selection-mutation balance, frequency dependent selection;
- See how these concepts can be formalized in a computer program;
- Appreciate both the power and the limits of natural selection.

2 Simulated Evolution

In the first part of this computer lab we will simulate the evolution of a (DNA) string of letters under a particular fitness function by using R.

Lets start by creating an individual of our population:

► First, start R or R-studio, depending on your operating system and preferences.

► Load the library `stringr` by typing `library(stringr)` in the command line.

► Generate a random string of length 10 containing the characters A,G,C and U. You can use the following command:

```
paste(sample(c('A','G','C','U'), size=10, replace=TRUE), collapse=")
```

which creates a vector of length 10 containing the letters 'A', 'G', 'C' and 'U' and them puts them all together in a string.

► Generate a couple of such strings to confirm that this does what you want (hint: you can use the arrow keys to scroll to commands you previously used in the command line).

► Now generate a random string of length 100 with the characters 'A', 'G', 'C' and 'U'. If you want, you can store the string under a name (for instance `x`), by typing `x <- paste(....'` (where the previous command goes on the dots).

⋆ How many such strings are possible? This is the genotype space.

Now, we want to create a population 100 of such strings:

► First create an empty vector to store your population: `population <- rep(0, 100)`

► Fill your vector by creating a string for every position:

```
for (i in 1:100) {
  population[i] <- paste(sample(c('A','G','C','U'), size=100, replace=TRUE),
collapse='')
}
```

Now we need to define a fitness function. Imagine, for instance, that the string CAC codes for a very useful aminoacids, such that the more CAC's in the genome, the higher the expected number of offspring. Thus, fitness = count(CAC).

► Create a vector containing the fitness of all the members of your population: first generate an empty vector to store the fitness' (`fitness <- rep(0, 100)`) and then use a for-loop to fill the vector with the fitness values, like in the previous bit of code. You can compute the fitness of an individual member of the population that is stored at place `i` in the population vector by using the function `str_count(`: the fitness of this member is `str_count(population[i], "CAC")`.

? What is the highest fitness a member of this population can have?

Now we will generate the next generation. Assume that each 'child' in the new population has a probability of inheriting their genome from a parent that is proportional to the parent's fitness. This is selection. (I think I need to formulate this a bit differently).

► Compute the average fitness and store it in a variable:

```
av_fitness <- mean(fitness)
```

► Generate 100 new children, using the built in function `sample` (the same one we used before):

```
population = sample(population, size=100, replace=TRUE, prob=100*av_fitness
```

? If one population member has fitness 20 and all the other population members have fitness 1, what is the probability that a child will inherit its genome from this one population member? What do you expect to happen with the population?

► Repeat this process 100 times and plot the result. If you feel like doing some implementation yourself, you can do this by creating a for-loop that executes the previous bits of codes 100 times, storing the fitness of every population in a vector. To plot your results, use:

```
plot(seq(1,100,1), av_fitness, type="l", ann=FALSE)
```

(Assuming you stored the fitness values in av_fitness).

To label your axes and titles you can use:

```
title(main="title", xlab="x label", ylab="y label")
```

Alternatively, you can use the provided script. Put your own values at the top!

⋆ Why does the fitness level off at a relatively low level?

Now lets introduce mutation: every child's nucleotide has a probability μ to change into a random nucleotide. TODO fix this function

▶ Use the provided script to do the same simulation, but with a mutation level $\mu = 0.01$. This is selection with mutation.

▶ Do 1000 repetitions with $\mu = 0.001$, plot the fitness. This shows the mutation-selection balance.

⋆ Why does the fitness with relatively high mutation rate level off at a slightly lower level?

3 Evolution of communication

A possible way of representing a communication system is by using matrices that describe a mapping from a set of meanings to a set of forms (or signals). For instance, the well known alarm call system of Vervet monkeys (Seyfarth et al., 1980) in its usual idealization, can be described as follows:

$$S = \left(\begin{array}{c|ccc} & \text{chip} & \text{grunt} & \text{chutter} \\ \hline \text{leopard} & 1 & 0 & 0 \\ \text{eagle} & 0 & 1 & 0 \\ \text{snake} & 0 & 0 & 1 \end{array}\right) \quad R = \left(\begin{array}{c|ccc} & \text{leopard} & \text{eagle} & \text{snake} \\ \hline \text{chirp} & 1 & 0 & 0 \\ \text{grunt} & 0 & 1 & 0 \\ \text{chutter} & 0 & 0 & 1 \end{array}\right)$$

The S matrix represents the sender: the first column contains the meanings (or situations) that the sender may want to express, the first row the signals that it can use to express these meanings. The numbers in the matrix represent the probabilities that the sender will use a certain signal to express a certain meaning. The matrix R describes the behaviour of the receiver in a similar way: the numbers in the matrix are the probabilities that the receiver will interpret a certain signal (first column) as having a certain meaning (first row).

More generally, if we have a set M with possible meanings and a set F with possible signals, then S is a $|M| \times |F|$ matrix that gives for every meaning $m \in M$ and signal $f \in F$ the probability that m is expressed with f. Similarly, R is a $|F| \times |M|$ matrix that gives for every $\langle f, m \rangle$ pair the probability that f is interpreted as m.

Explain how to compute how successful communication is?

⋆ What are the optimal S* and R*, for maximal communicative success in a population?

To study the evolution of such a communication system, we can use the same protocol as in the previous part of this assignment. Assume that every individual is characterized by a genome of length 18, where each nucleotide codes for one value in S and R. Let's say A=3, G=2, C=1 and U=0. To construct the S and R matrices, the rows need to be normalised.

? What would be a genome corresponding to the S and R matrix depicted above?

? Can you think of two strings that have a different genotype but the same phenotype?

Of course our previous fitness function - the count of the substring "CAC" - does not make much sense in this case. We will have to define a new one. When communicating with a fixed target (thus constant S and R matrices), the chance of successful communication can be computed by ... (why is it this way).

We provided some implemented fitness functions in the file fitness_functions.R:

- `CAC_count`: This is the fitness function you used before, that counts the number of occurrences of the substring "CAC" in the genome;
- `communication_fixed_target`: This fitness function captures how well the population member can communicate with a fixed target with S and R matrix that allows perfect communication (i.e., it does not use the same signal for different meanings, or assign different meanings to the same signal).
- `communication_random_target`: This fitness function describes the more realistic situation, in which the fitness of a population member is determined based on its communication with a random other member of the population.

You can change the fitness function of your simulation by commenting out the previous fitness function (in this case the line that says '`fitness_function <- CAC_count`') and uncommenting the line with the preferred fitness function. As you may have guessed, you can (un)comment a line in an R script by placing (removing) a '#' at the beginning.

? What is the maximal fitness that an individual can have?

▶ Change the fitness function in the script to `communication_fixed_target`. Run an evolutionary simulation with a low mutation rate for 100 iterations. What is the average fitness and most frequent communication system at the end of it? Is it trivial how they can check the population at the end? Maybe make something to do that

? Can the members of the resulting population also communicate with each other or only with the preset fixed target?

? What would happen if the target was fixed, but not perfect? You can test your assumption by changing the target matrices in the fitness_functions file.

A more realistic situation is the one in which the members of the population do not all communicate with the same fixed target, but with other members of the population, that has his own (evolved) S and R matrix.

▶ Run some evolutionary simulations for this scenario (compute the fitness by using the function `communication_random_target`). What is the average fitness and most frequent communication system at the end of it?

⋆ This is frequency dependent selection. Why does it not always evolve to the optimal communication system?

- Compute fitness by communicating with a a fixed target S and R. Fitness = sum of diagonal values of S%*%R* + S*%*%R.
- Run an evolutionary simulation with low mutation rate for 100 iteration. What is the average fitness and most frequent communication system at the end of it?
- Compute fitness by communicating with a a random other agent, with its own (evolved) S' and R'. Fitness = sum of diagonal values of S%*%R' + S'%*%R.
- Run an evolutionary simulation with low mutation rate for 100 iteration. What is the average fitness and most frequent communication system at the end of it?
- *This is frequency dependent selection. Why does it not always evolve to the optimal communication system?

TODO: say something about running scripts from R (folder, `source`)

TODO: explain download fitness function

4 Communication Systems as Matrices

We don't need this anymore I think?

Campbell monkeys have an alarm call system where the calls for leopards and eagles can be preceded by a "boom" call, which generally has the effect of changing the meaning of the calls from predator-specific alarms to a general signal of disturbance, although they are sometimes still interpreted as alarms (Zuberbühler, 2002). If we consider just the calls for leopards and eagles, with and without preceding boom, we have 4 different signals and, if we add "disturbance" a set of 3 different meanings.

You can compute the chance of successful communication by summing up the chance of success for each individual meaning-signal combination.For instance, let's assume the sender wants to convey the meaning "leopard". We multiply the probabilities for all signals the sender could use for this meaning (the row in S starting with "leopard") with the chance that the receiver will interpret this signal as having the meaning "leopard" (the "leopard" column in R). In this case as the sender only uses the signal L$_{\text{C}}$ to express the meaning leopard, and the receiver interprets this signal as having the meaning leopard with probability 1, the chance of successfully conveying the meaning "leopard" in this system is 1.

References

Robert M Seyfarth, Dorothy L Cheney, and Peter Marler. Monkey responses to three different alarm calls: evidence of predator classification and semantic communication. *Science*, 210(4471):801–803, 1980.

Klaus Zuberbühler. A syntactic rule in forest monkey communication. *Animal behaviour*, 63(2):293–299, 2002.