Homework 4: Genetic algorithm functions I

Dylan Schwilk

Due date: September 25, 2018

How to represent simplified genotypes

In genetic algorithms or in simulating organic evolution, it is common to represent a simplified "genotype" as a vector of numbers, characters, or bits (1s and 0s or TRUEs and FALSEs).

We will start building a system for simulating the evolution of such vectors which will represent haploid genotypes. One of the first functions we need for such a program is a function that takes two parents and produces an offspring. In our first version of this function, we will ignore mutation, but we will be concerned with recombination. The function will have the following interface:

```
# mateVectors(x, y, r):
# Returns a child vector created by "mating" two parent
# vectors x and y and supplying a per-adjacent-locus recombination rate.
#
# Args:
# - x, y = the two parent vectors. These must be of same
# type (numeric, character, or logical) and same
# length.
# - r = the per-locus recombination rate
#
# Returns:
# - A vector of the same length as the parents
```

An outline of the algorithm

We are simulating a simplified haploid recombination mechanism. This is not meant to directly represent biology but to provide a way to simulate mating with limited recombination and allow spatial correlation such that adjecent alleles are more likely to stay together. Your function will use the following algorithm for mating and recombination:

- Randomly select one parent from which to begin reading values and transferring these values (alleles) to offspring
- After each vector element is read and the result copied to the child vector, check for a crossing over event (with probability r)
- On a recombination event, a crossover occurs and the reading then switches to the second parent.

- When length(parent) values have been transferred to the child, the process is complete.

It is possible to implement this function either with explicit loops (which is how I've described the algorithm, above) or with vectorized calls.

Here is an example use of the function you will create:

```
mother <- rep(0,50)
father <- rep(1,50)
mateVectors(mother,father, 0.02)
## [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0
##[39] 0 0 0 0 0 0 0 0 0 0 0 0
```

And another example using strings instead of integers:

```
mother <- unlist(strsplit("I am the mother", split=""))
father <- unlist(strsplit("i AM THE FATHER", split=""))
child <- mateVectors(mother,father, 0.05)
paste(child,collapse="")
## [1] "I am the mothER"
```