Population Genomics to Adaptation

In this practical session we will simulate a number of different evolutionary scenarios and we will see the different patterns by studying a number of summary statistics: The variability (Theta=4Neμ), The Site Frequency Spectrum (SFS) and the estimation of the proportion of adaptive substitutions (alpha).

1. We will simulate a number of different scenarios using ***Slim*** (Messer, Genetics 2013, Haller and Messer, MBE 2017). Slim is a forward simulator that allows to simulate many selective positions at the same time in complex demographic patterns. Slim2 has a graphical interface for MacOSX but here, we will use the **command line program** for the practical session.

> **1. SLiM overview**
>
> **1.1 Introduction**
>
> SLiM is an evolutionary simulation package that provides facilities for very easily and quickly constructing genetically explicit individual-based evolutionary models. At the simplest level SLiM is based upon a Wright-Fisher-type model of evolution; in particular, (1) generations are non-overlapping and discrete, (2) the probability of an individual being chosen as a parent for a child in the next generation is proportional to the individual's fitness, (3) individuals are diploid, and (4) offspring are generated by recombination of parental chromosomes with the addition of new mutations. Some of these assumptions can be relaxed using techniques described in this manual; nevertheless, this is the foundation upon which all more complex SLiM models are built, and when these assumptions are relaxed, it is by explicitly modifying this base behavior.

(*Slim* manual)

2. In order to run Slim, we need to construct the scripts with the models to study. We will use a **template** that we will modify in relation to the different scenarios, with the help of the *Slim* manual. This template contains orders to run a simulation and extract a sample in ***ms*** format.

    a. We will simulate a 30Kb DNA fragment of a coding region of an initial population of 10K individuals.

```
// set up a simple neutral scenario simulation
initialize() {

  // set the overall uniform mutation rate
  initializeMutationRate(5e-6);
  // uniform recombination along the chromosome
  initializeRecombinationRate(5e-6);

  // m1 mutation type: (neutral)
  initializeMutationType("m1", 0.5, "f", 0.0);
  // m2 mutation type: (deleterious)
  initializeMutationType("m2", 0.5, "f", -1.0);
  // m3 mutation type: (beneficial)
  initializeMutationType("m3", 0.5, "f", 0.1);

  // g1 genomic element type: (synonymous) uses m1 for all mutations
  initializeGenomicElementType("g1", m1, 1.0);
  // g2 genomic element type: (nonsynonymous) uses all mutations
  initializeGenomicElementType("g2", c(m1,m2,m3), c(1,0,0));

  // Define positions to genomic elements at a chromosome of length 30 kb
  count = 0;
  for (i in 0:30000) {
       count2 = count;
       count = count2 + 1;
       if (count == 1) {
            initializeGenomicElement(g1, i, i);
       } else if (count == 2) {
            initializeGenomicElement(g2, i, i);
       } else if (count == 3) {
            initializeGenomicElement(g2, i, i);
            count = 0;
       }
  }

  //in case beneficial selection at a single position in the middle of chromosome
}
```

```
// create a population of 1000 individuals
1{
  sim.addSubpop("p1", 1000);
}

// Split de p1 in the generation 10000.
// Now we have the target pop and the outgroup pop.
10000 { sim.addSubpopSplit("p2", 1000, p1); }

// If required, sudden change in population size
28000 { p2.setSubpopulationSize(1000); }

//if required, force a strong selective sweep
//29800 {
//     target = sample(p2.genomes, 100); //the mutation is in 100 individuals
//     target.addNewDrawnMutation(m3, 15000); //the mutation occurs at position 15000
//}

// Run to the final
30000 late() {
  // Select 100 samples from the 1000 individuals and output to MS format
  // Select 10 samples of the outgroup and 100 samples of the target population and output
to MS format
  // obtain random samples of genomes from the three subpopulations
  g_1 = sample(p2.genomes,100,F);
  g_2 = sample(p1.genomes,10,F);
  //Concatenate the 2 population samples
  g_12=c(g_1,g_2);
       //Get the unique mutations in the sample, sorted by position
       m = sortBy(unique(g_12.mutations),"position");
       // print the number of segregating sites
       cat("//" + "\n");
       cat("segsites: "+ size(m) + "\n");
       //print the positions
       positions = format("%.6f", m.position / sim.chromosome.lastPosition);
       cat("positions: "+ paste(positions," ") + "\n");
       //print the sampled genomes
       for (genome in g_12){
            hasMuts = (match(m,genome.mutations) >= 0);
            cat(paste(asInteger(hasMuts),"") + "\n");
       }
}
```

3. The scenarios are the following (12 scenarios, we will divide the task in 12 groups):

   a. A two species model **with constant population size** and **no selection** on functional positions. **No recombination** between positions.

   ```
   // uniform recombination along the chromosome
   initializeRecombinationRate(0.0);
   ```

   b. A two species model with constant population size and no selection on functional positions. **High recombination** between positions.

   ```
   // uniform recombination along the chromosome
   initializeRecombinationRate(5e-6);
   ```

   c. A two species model with constant population size plus recent **strong positive selection on a single** position in the target population. No recombination between positions.

   ```
   // uniform recombination along the chromosome
   initializeRecombinationRate(0.0);

   //if required, force a strong selective sweep
   29800 {
        target = sample(p2.genomes, 100); //the mutation is in 100 individuals
        target.addNewDrawnMutation(m3, 15000); //the mutation occurs at position 15000
   }
   ```

   d. A two species model with constant population size plus recent strong positive selection on a single position in the target population. **High recombination** between positions.

   ```
   // uniform recombination along the chromosome
   initializeRecombinationRate(5e-6);

   //if required, force a strong selective sweep
   29800 {
        target = sample(p2.genomes, 100); //the mutation is in 100 individuals
        target.addNewDrawnMutation(m3, 15000); //the mutation occurs at position 15000
   }
   ```

e. A two species model with constant population size plus **beneficial selection (low proportion, 5%)** on functional positions in the target population. High recombination between positions.

```
// g2 genomic element type: (nonsynonymous) uses all mutations
initializeGenomicElementType("g2", c(m1,m2,m3), c(1,0,0.05));
```

f. A two species model with constant population size plus **beneficial selection (middle proportion 15%)** on functional positions in the target population. High recombination between positions.

```
// g2 genomic element type: (nonsynonymous) uses all mutations
initializeGenomicElementType("g2", c(m1,m2,m3), c(1,0,0.15));
```

A two species model with constant population size plus **beneficial selection (high proportion 50%)** on functional positions in the target population. High recombination between positions.

```
// g2 genomic element type: (nonsynonymous) uses all mutations
initializeGenomicElementType("g2", c(m1,m2,m3), c(1,0,0.5));
```

g. A two species model, **demographic expansion (5x)** in the target population and no selection on functional positions in the target population. High recombination between positions.

```
// If required, sudden change in population size
28000 { p2.setSubpopulationSize(5000); }
```

A two species model, **demographic reduction (0.2x)** in the target population and **no selection** on functional position in the target population s. High recombination between positions.

```
// If required, sudden change in population size
28000 { p2.setSubpopulationSize(200); }
```

h. A two species model with constant population size and **deleterious mutations** on functional positions in the target population. High recombination between positions.

```
// g2 genomic element type: (nonsynonymous) uses all mutations
initializeGenomicElementType("g2", c(m1,m2,m3), c(1,9,0));
```

i. A two species model, **demographic expansion** in the target population and **deleterious mutations** on functional positions in the target population. High recombination between positions.

```
// g2 genomic element type: (nonsynonymous) uses all mutations
initializeGenomicElementType("g2", c(m1,m2,m3), c(1,9,0));

// If required, sudden change in population size
28000 { p2.setSubpopulationSize(5000); }
```

j. A two species model, demographic expansion in the target population, deleterious **and beneficial selection (low proportion, 5%)** on functional positions in the target population. High recombination between positions.

```
// g2 genomic element type: (nonsynonymous) uses all mutations
initializeGenomicElementType("g2", c(m1,m2,m3), c(1,9,0.05));
```

k. A two species model, demographic expansion in the target population, deleterious **and beneficial selection (middle proportion, 15%)** on functional positions in the target population. High recombination between positions.

```
// g2 genomic element type: (nonsynonymous) uses all mutations
initializeGenomicElementType("g2", c(m1,m2,m3), c(1,9,0.15));
```

l. A two species model, demographic expansion in the target population, deleterious **and beneficial selection (high proportion, 50%)** on functional positions in the target population. High recombination between positions.

```
// g2 genomic element type: (nonsynonymous) uses all mutations
initializeGenomicElementType("g2", c(m1,m2,m3), c(1,9,0.5));
```

4. Once we run the scripts in *Slim* and the output is obtained, we need to obtain the summary statistics that explain the patterns of variability.
   a. First cut the header of the output and keep only the *ms* simulation:
      i. sed '1,/Starting run/d'< *filename_output_slim* > *file_name_ms*
   b. We will use *mstatspop* program to calculate the variability across the genome (theta Watterson) and the Site Frequency Spectrum (SFS) for neutral (synonymous) and functional positions (nonsynonymous).
      i. You need to introduce a *mask file* to separate neutral from functional positions. Therefore, the program must be run twice: first for neutral and next for functional positions. The command line to run *mstatspop* (https://github.com/CRAGENOMICA/mstatspop) using the *ms* format as input is the following:
         1. *mstatspop -f ms -i file_name_ms -o 0 -N 2 100 10 -G 1 -l 30000 -r 1 -n name_scaffold.txt -m mask_neutral.txt > file_name_neutral_statistics.txt*
         2. *mstatspop -f ms -i file_name_ms -o 0 -N 2 100 10 -G 1 -l 30000 -r 1 -n name_scaffold.txt -m mask_functional.txt > file_name_functional_statistics.txt*
      ii. The (variability levels and Site Frequency Spectrum is available inside the output file:
         1. Example:

(…)
Estimates of variability for each population (an and bn for the variant positions):
S[0]: 23 **Theta(Wat)[0]: 4.442403** Theta(Taj)[0]: 3.378182
Theta(Fu&Li)[0]: 3.000000 Theta(Fay&Wu)[0]: 1.389495
Theta(Zeng)[0]: 2.383838 Theta(Achaz,Wat)[0]: 4.787693
Theta(Achaz,Taj)[0]: 3.385900 **Divergence[0]: 91.660000**
Estimates of NUCLEOTIDE variability for each population (if missing, corrected by the averaged effective positions):
S[0]: 23 Theta/nt(Wat)[0]: 0.000444 Theta/nt(Taj)[0]: 0.000338
Theta/nt(Fu&Li)[0]: 0.000300 Theta/nt(Fay&Wu)[0]: 0.000139
Theta/nt(Zeng)[0]: 0.000238 Theta/nt(Achaz,Wat)[0]: 0.000479
Theta/nt(Achaz,Taj)[0]: 0.000339 Divergence[0]: 0.009166
(…)
Frequency of variants for each population:
fr[0,1]: 3 fr[0,2]: 5 fr[0,3]: 2 fr[0,4]: 2 fr[0,5]: 2 fr[0,6]: 2 fr[0,7]: 0 fr[0,8]: 0
fr[0,9]: 1 fr[0,10]: 1 fr[0,11]: 0 fr[0,12]: 0 fr[0,13]: 0 fr[0,14]: 0 fr[0,15]: 0
fr[0,16]: 1 fr[0,17]: 0 fr[0,18]: 0 fr[0,19]: 0 fr[0,20]: 0 fr[0,21]: 0 fr[0,22]: 0
fr[0,23]: 0 fr[0,24]: 0 fr[0,25]: 0 fr[0,26]: 1 fr[0,27]: 0 fr[0,28]: 0 fr[0,29]: 1
fr[0,30]: 0 fr[0,31]: 0 fr[0,32]: 0 fr[0,33]: 0 fr[0,34]: 0 fr[0,35]: 0 fr[0,36]: 0
fr[0,37]: 0 fr[0,38]: 0 fr[0,39]: 0 fr[0,40]: 0 fr[0,41]: 0 fr[0,42]: 0 fr[0,43]: 0
fr[0,44]: 0 fr[0,45]: 1 fr[0,46]: 0 fr[0,47]: 0 fr[0,48]: 0 fr[0,49]: 0 fr[0,50]: 0
fr[0,51]: 0 fr[0,52]: 1 fr[0,53]: 0 fr[0,54]: 0 fr[0,55]: 0 fr[0,56]: 0 fr[0,57]: 0
fr[0,58]: 0 fr[0,59]: 0 fr[0,60]: 0 fr[0,61]: 0 fr[0,62]: 0 fr[0,63]: 0 fr[0,64]: 0
fr[0,65]: 0 fr[0,66]: 0 fr[0,67]: 0 fr[0,68]: 0 fr[0,69]: 0 fr[0,70]: 0 fr[0,71]: 0
fr[0,72]: 0 fr[0,73]: 0 fr[0,74]: 0 fr[0,75]: 0 fr[0,76]: 0 fr[0,77]: 0 fr[0,78]: 0
fr[0,79]: 0 fr[0,80]: 0 fr[0,81]: 0 fr[0,82]: 0 fr[0,83]: 0 fr[0,84]: 0 fr[0,85]: 0
fr[0,86]: 0 fr[0,87]: 0 fr[0,88]: 0 fr[0,89]: 0 fr[0,90]: 0 fr[0,91]: 0 fr[0,92]: 0
fr[0,93]: 0 fr[0,94]: 0 fr[0,95]: 0 fr[0,96]: 0 fr[0,97]: 0 fr[0,98]: 0 fr[0,99]: 0
(…)

      iii. Extracting the levels of variability, the divergence and the site frequency spectrum for neutral and functional positions. For SFS:
         1. *for i in `seq 1 99`; do answer=$(echo "scale=2; $i /100;" | bc); echo "$answer" >> freq_col.txt; done #create a column with frequencies*
         2. *grep '\fr\[0,' file_name_functional_statistics.txt | tr '\t' '\n' | cut -d ' ' -f2 > file_name_functional_statistics_SFS.txt #create a column with functional SFS.*

3. *grep 'fr\[0,' file_name_neutral_statistics.txt | tr '\t' '\n' | cut -d ' ' -f2 > file_name_neutral_statistics_SFS.txt #create a column with neutral SFS.*
4. *paste freq_col.txt file_name_functional_statistics_SFS.txt file_name_neutral_statistics_SFS.txt >> file_name_cols.txt #join the frequencies and the SFS for functional and neutral.*
5. *echo "freq\tsfs_funct\tsfs_neutral" > file_name_SFS.txt #make a header.*
6. *cat file_name_cols.txt >> file_name_SFS.txt #join header with SFS columns.*

c. We will use the asymptotic MKT approach to estimate the proportion of adaptive substitutions. (http://benhaller.com/messerlab/asymptoticMK.html, from Haller and Meeser G3 2017).
   i. It is necessary to introduce the functional and neutral divergence (from mstatspop file) and the file with a table containing the three columns (the file previously constructed): the frequency of the derived allele, the number of functional variants for each frequency and the number of neutral variants for each frequency.
   ii. This algorithm estimates the value of α for each frequency separately and make a plot. Finally, estimate asymptotically the value of α from the whole data, to eliminate the effect of deleterious mutations.

5. Finally, we will evaluate the results:
   a. Observe the level of variability in relation to the value included in the simulation (Theta=4Neμ).
   b. The comparison of the Site frequency Spectrum between neutral and functional positions. (make 2 plots, with neutral and functional).
   c. The estimation of the proportion of adaptive substitutions in relation to the value included in the simulation (using asymptotic MKT approach). Take the raw value, the asymptotic value and the plot.
   d. Comparison of the results from all studied conditions. Make a table with the results given the simulated conditions, Interpretation.