FANTASTIC FORCES

5th - 7th June, 2019
University of St Andrews, Scotland

How to detect the true targets of selection in Experimental Evolution?

Carolina Barata

2019 June

1 How to simulate allele frequency trajectories?

You must have the slightest hint of what we are about to spend the next 2 hours on, right? Absolutely, we will be investigating genomic patterns of short-term adaptation! Too enthusiastic? Don't worry, you too will be buzzing once you have learned how to estimate selection parameters from time series polymorphism data.

Most evolution experiments in the laboratory investigate both phenotypic and genomic responses to some induced selective pressure. Also, it is common practice to sequence pools of individuals to accurately estimate population allele frequencies (while trying not to blow your lab budget). So, our aim is to get you acquainted with time series polymorphism data and one of the many population genetics inference methods for experimental evolution.

We will start by simulating 50 allele frequency trajectories. Then, we will plot these frequency data across 125 generations of evolution, and estimate selection coefficients for each of these alleles.

Our simulating function uses the Moran model of nucleotide evolution to describe an allele frequency trajectory. The Moran model is simply a birth-death process, which assumes overlapping generations, continuous time and discrete allele frequency states.

For this tutorial, we will be using R to perform simulations and run analyses. You can download BAIT-ER from `https://github.com/mrborges23/Bait-ER`.

Let's start by defining some key experimental parameters:

```
# Experimental parameters
ne <- 500 # effective population size
sampling_scheme <- seq(0, ne/4, length.out = 6) # time points
replicates <- 5 # number of experimental populations
coverage <- 80 # total sequencing depth at the site
sigma <- 0 # true selection coefficient
```

```
initial_frequency <- 0.1 # starting allele frequency
sites <- 50 # number of trajectories to simulate

# Initial Moran states
prob_vector_initial <- rep(0, ne + 1) # state space: from 0 to Ne copies
prob_vector_initial[floor(initial_frequency * ne) + 1] <- 1
```

To simulate 50 neutral allele frequency trajectories, we use the function allele_frequency_simulator(), which requires 6 arguments:

- Effective population size, N_e;
- Simulated (true) selection coefficient, sigma;
- Sampling scheme, i.e. list of time points for which allele frequencies should be simulated;
- List of initial state probabilities;
- Number of replicate populations;
- And total allele coverage.

```
all_neutral_trajectories <- list()
for (position in 1:sites) {
  reads <- allele_trajectory_simulator(ne,
                                       sigma,
                                       sampling_scheme,
                                       prob_vector_initial,
                                       replicates,
                                       coverage)
    all_neutral_trajectories[[position]] <- reads
}
```

Now, let's simulate a scenario where directional selection is strong:

```
sigma <- 10/ne # set sigma to something different from 0
all_selected_trajectories <- list()
for (position in 1:sites) {
  reads <- allele_trajectory_simulator(ne,
                                       sigma,
                                       sampling_scheme,
                                       prob_vector_initial,
                                       replicates,
                                       coverage)
    all_selected_trajectories[[position]] <- reads
}
```

Finally,onto plotting the simulated data over time:

```
# Compute allele frequencies
neutral_freq <- matrix(sapply(all_neutral_trajectories,
                              function(x)
                                {x$allele_coverage/coverage}),
                       ncol = length(sampling_scheme),
                       byrow = TRUE)
selected_freq <- matrix(sapply(all_selected_trajectories,
                               function(x)
```

```
                                {x$allele_coverage/coverage}),
                        ncol = length(sampling_scheme),
                        byrow = TRUE)

# Plot 10 trajectories
par(mfrow=c(2,1))
plot(x = rep(sampling_scheme, 10),
     y = neutral_freq[sample(nrow(neutral_freq), 10),],
     type = "l",
     ylim = c(0,1),
     xlab = "Generations",
     ylab = "Observed allele frequency",
     main = "Neutral evolution",
     col = topo.colors(10))
plot(x = rep(sampling_scheme, 10),
     y = selected_freq[sample(nrow(selected_freq), 10),],
     type = "l",
     ylim = c(0,1),
     xlab = "Generations",
     ylab = "Observed allele frequency",
     main = "Strong selection",
     col = topo.colors(10))
```

2 How to estimate selection coefficients?

BAIT-ER takes all replicate populations into account when computing the likelihood of observing the data given some selection coefficient, sigma. It, thus, include all available data into the model to make use of consistency among allele frequency trajectories. This model also accounts for varying coverage at a given loci by

We will be using the BAIT-ER sigma_posterior() function, which outputs a point estimate of sigma and its posterior distribution.

```
neutral_sigmas <- sapply(all_neutral_trajectories,
                         function(x)
                           sigma_posterior(N = ne,
                                           time = sampling_scheme,
                                           number_time_points = length(sampling_scheme),
                                           number_replicates = replicates,
                                           trajectories_matrix = reads_to_moran_states(ne, x),
                                           prior_parameters = c(0.001,0.001))$mean)

selected_sigmas <- sapply(all_selected_trajectories,
                          function(x)
                            sigma_posterior(N = ne,
                                            time = sampling_scheme,
                                            number_time_points = length(sampling_scheme),
                                            number_replicates = replicates,
                                            trajectories_matrix = reads_to_moran_states(ne, x),
                                            prior_parameters = c(0.001,0.001))$mean)
```

Let us now produce histograms of these mean sigma estimates for both scenarios:

```
par(mfrow=c(1,1))
hist(neutral_sigmas * ne,
     xlim = c(floor(min(neutral_sigmas * ne)),
```

```
              ceiling(max(selected_sigmas * ne))),
     breaks = 10,
     col = rgb(205, 181, 205, alpha = 255, max = 255),
     xlab = expression(paste("Scaled ", hat(sigma), " BAIT-ER")),
     ylab = "Counts",
     main = "Estimated selection coefficients")
hist(selected_sigmas * ne, add = T, breaks = 10,
     col = rgb(0, 139, 139, alpha = 150, max = 255))
abline(v = 0, col="thistle3", lwd=3, lty=2)
abline(v = 10, col = "cyan4", lwd=3, lty=2)
```

3 How to detect selection in real data?

For the second part of our tutorial, we will be analising a published dataset by Barghi *et al*, 2019. The authors established 10 replicate *Drosophila simulans* laboratory populations to a new temperature regime. The aim of this experiment was to find genomic signatures of adaptation to a novel environment across replicates for 60 generations of evolution. The estimated effective population size is approximately 300 individuals, despite the fact that the census population size was kept constant throughout the experiment at 1000 individuals.

Often, pooled sequencing experiment data are stored as .sync files for convenience. Most experimental evolution bioinformatics tools for estimating selection parameters allow for this specific input format. The first 3 fields correspond to:

- Reference contig/chromosome;
- Position within the reference contig/chromosome;
- Reference allele/character.

The remaining fields show the allele counts for all the replicate populations and time points.

We will be analysing a subset of the original dataset, focusing on the right arm of the X chromosome. Let's start by looking at how the .sync file is structured:

```
# Print first line of sync file
sync_file <- "Dsim_F0-F60_Q20_polymorphic_2L_20ksnp_subset.sync"
read.table(file = sync_file, sep = "\t")[1,]

# Experimental parameters
sites <- 1000
replicates <- 10
time_points <- c(0, 10, 20, 30, 40, 50, 60)
estimated_n <- 300

# Read in sync file
sync_data <- read_sync_file(path = sync_file,
                            number_sites = sites,
                            number_time_points = length(time_points),
                            number_replicates = replicates)
```

Now that we have read in the file, we can estimate selection coefficients for this subset of SNPs:

```
# Estimate posterior sigmas
sites_sigma_posterior(sync_file = sync_data,
                      initial_site = 1,
                      final_site = sites,
                      output_file = paste0(sync_file, ".baiter"),
                      N = estimated_n,
```

```
                    time = time_points,
                    number_time_points = length(time_points),
                    number_replicates = replicates,
                    prior_parameters = c(1, 1))
```

Lastly, we will be plotting the estimated selection coefficients across this genomic region:

```
# Read in BAIT-ER output file
baiter_est <- read.table(file = paste0(sync_file, ".baiter"))
              # V2 = genomic position, V6 = estimated sigma

# Plot estimated sigmas along the genomic region
plot(x = baiter_est$V2,
     y = baiter_est$V6 * estimated_n,
     col = ifelse(baiter_est$V6 * estimated_n > 10,
                  "cyan4", "gray17"),
     pch = 19,
     ylab = expression(paste("Scaled ", hat(sigma), " BAIT-ER")),
     xlab = "Genomic position (bp)",
     main = "Barghi et al (2019) 2L subset")

abline(h = 10, col = "cyan4", lty = 2) # add threshold
```

And that's a wrap!

Data reference N. Barghi, V. Nolte, T. Taus, A. M. Jaksic, R. Tobler, F. Mallard, M. Dolezal, C. Schlotterer, R. Kofler, and K. A. Otte. Genetic redundancy fuels polygenic adaptation in Drosophila. *PLOS Biology*, 17(2):e3000128, 2019.