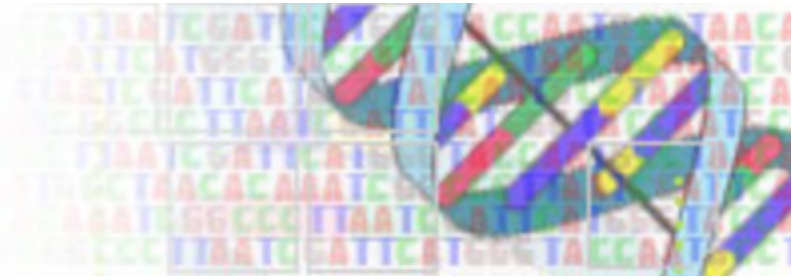
THE UNIVERSITY OF MELBOURNE

Faculty of Sciences
School of BioSciences



MSc - Bioinformatics
Bioinformatics Case Studies
BINF90004

Genetic Adaptation in the face of Environmental Change

12 August 2024

Alex Fournier-Level

afournier@unimelb.edu.au

adaptive-evolution.org

URL to project

Objectives of this lab class

- Test the effect of different **genetic architectures** on the capacity of a population to 1- **adapt** and 2- **overcome environmental change**.
    - *What do we mean by genetic architecture?*

Objectives of this lab class

- Test the effect of different **genetic architectures** on the capacity of a population to 1- **adapt** and 2- **overcome environmental change**.
    - *What do we mean by genetic architecture?*
    - How much of the variation is genetic?
    - How many genes?
    - How big are the genes effects?

Using the good old quantitative genetics framework

- Use of a forward simulation framework (`>Selection.Simulator()`)
- Relying on a “usual” linear/additive decomposition of the variance
  - Y = Genetics (G) + Environment (E)
- With $G \sim N(0 ; \sigma_G^2)$ and $E \sim N(0 ; \sigma_E^2)$
- G is the sum of `LociNum` individual gene effects, either binomial (“`Equal Loci`”) or gamma distributed (“`continuous`”)

Working in the R environment

- First, simulate data following a defined set of parameters using functions of the native R environment (`Base package`)
  Random draws (`sample()`, `rnorm()`, `rgamma()`)

- Estimate parameters of interest using linear models
  - linear mixed-modeling (`lme4 package`)
    - mean and variance for the trait
    - variance components of the model and **heritability**

How to install a package

- From CRAN:
- `>install.packages("lme4")`
- From source:
- `>install.packages("lme4-XXXX.tar.bz2",repos=NULL,type="source")`
- You may also want to install "`compiler`", which is native but sometimes goes missing...

Let's get this started

- Download the *Dynamic_Selection_beta.r* script available at:
  - https://github.com/aflevel/Dynamic_Selection
- Open it with you favorite text editor, or R Studio if that's what you want... and run everything
- Let's have a look at the arguments of the function

Let's get this started

- Download the *Dynamic_Selection_beta.r* script available at:
  - https://github.com/aflevel/Dynamic_Selection
- Open it with you favorite text editor, or R Studio if that's what you want... and run everything
- Let's have a look at the arguments of the function
- Run the function `>Selection.Simulator()`
  - *Can everybody see a distribution histogram?*

Let's get this started

- Download the *Dynamic_Selection_beta.r* script available at:
  - https://github.com/aflevel/Dynamic_Selection
- Open it with you favorite text editor, or R Studio if that's what you want... and run everything
- Let's have a look at the arguments of the function
- Run the function `>Selection.Simulator()`
- Now run `>Selection.Simulator()`
  - with different environmental variance values
    - `>sdEnvironmentalFX=0.01`
    - `>sdEnvironmentalFX=0.1`
    - `>sdEnvironmentalFX=1`
    - *What do you reckon?*

Let's get this started

- **First key result:**
  - It doesn't take much environmental variance to make a genetically binomial trait look continuous!

Focusing on Heritability

$$Y = \mu + \underbrace{G}_{\text{Genotype}} + \underbrace{E}_{\text{Environment}}$$

$$\sigma^2_Y = \sigma^2_G + \sigma^2_E$$

Broad-sense heritability: $H^2_Y = \sigma^2_G / \sigma^2_Y$

- *Print out the heritability from* `Selection.Simulator()`

Focusing on Heritability

$$Y = \mu + \underbrace{G}_{\text{Genotype}} + \underbrace{E}_{\text{Environment}}$$

$$\sigma^2_Y = \sigma^2_G + \sigma^2_E$$

Broad-sense heritability: $H^2_Y = \sigma^2_G / \sigma^2_Y$

- *Print out the heritability from* `Selection.Simulator()`
  - `>Selection.Simulator()[[1]][1,1]`

Focusing on Heritability

- *Print out the heritability from* `Selection.Simulator()`
  - `>Selection.Simulator()[[1]][1,1]`
- *Print out the heritability for 100 runs of* `Selection.Simulator()` *and store the results in a vector*
  - `>Graph=FALSE`

Focusing on Heritability

- *Print out the heritability from* `Selection.Simulator()`
  - `>Selection.Simulator()[[1]][1,1]`
- *Print out the heritability for 100 runs of* `Selection.Simulator()` *and store the results in a vector*
  - `>Graph=FALSE`
  - `>heri.serie=vector()`
  - `>for (i in 1:100)`
    `heri.serie=c(heri.serie,Selection.Simulator()`
    `[[1]][1,1])`
  - `>hist(heri.serie)`
  - `>mean(heri.serie)`

Focusing on Heritability

- *Is this the best you can do?*

Focusing on Heritability

- *Is this the best you can do?*
  - Of course not!
  - The `mean()` estimate of a variable that is not symetrically distributed is biased (here inflated).
- Let's use maximum likelihood instead!
  - First write the log-likelihood function as
    - `function(par,Y) {sum(pdf(Y1,Y2.../par))}`
  - where par are the parameters of the probability density function (pdf) of a chosen distribution.
  - *try norm and gamma distributions*
    - `>dnorm(y, mean, sd, log=T)`
    - `>dgamma(y, rate, shape, log=T)`

Focusing on Heritability

- First write the log-likelihood function as
  - `function(par,Y) {sum(pdf(Y1,Y2.../par))}`
  - where par are the parameters of the probability density function (pdf) of a chosen distribution
- Second, find the `par` that maximise this function using
  - `>optim(par, fn, y)`

Focusing on Heritability

- First write the log-likelihood function as
  - `>function(par,Y) {sum(pdf(Y1,Y2.../par))}`
  - where par are the parameters of the probability density function (pdf) of a chosen distribution
- Second, find the `par` that maximise this function using
  - `>optim(par, fn, y)`
- And get the parameter estimate with
  - `>optim()$par`

Focusing on Heritability

- Here is the solution

- for a normal
- `>loglik.norm <- function(par, y) {sum(dnorm(y, mean=par[1], sd=sqrt(par[2]), log = TRUE))}`
- `>MLest.norm <- optim(c(mean = 0, var = 1), fn = loglik.norm,y = heri.seri, control = list(fnscale = -1,reltol = 1e-16))$par`
- and for a gamma distribution
- `>loglik.gamma <- function(par, y) {sum(dgamma(y,shape=par[1],rate=par[2],log=TRUE))}`
- `>MLest.gamma <- optim(c(shape = 1, rate = 1), fn = loglik.gamma,y = heri.serie, control = list(fnscale = -1,reltol = 1e-16))$par`
- and print the actual likelihood of the model with
- `>loglik.norm(MLest.norm,heri.serie)`
- `>loglik.gamma(MLest.gamma,heri.serie)`

Focusing on Heritability

- However, the best is the lognormal distribution

- ```
  >loglik.lognorm <- function(par, y)
  {sum(dnorm(log(1+y), mean=par[1], sd=sqrt(par[2]),
  log = TRUE))}
  ```
- ```
  >MLest.lognorm <- optim(c(mean = 0, var = 1), fn =
  loglik.lognorm,y = heri.serie, control = list(fnscale
  = -1,reltol = 1e-16))$par
  ```

Focusing on Heritability

- Alright, true if only one gene is involved (LociNum=1),
- but is it always true?

- Can you plot the distribution histogram of the heritability estimates
- `>for (LociNum in c(1,5,10,50,100)) {}`

- First create
- `>heri.esti=list()`

- where you going to store the results as
- `>heri.esti[[paste(LociNum)]][[norm]]=MLest.norm`

  - and for gamma: `mean=shape/rate`
  - `var=shape/rate^2`

- And print the likelihood of each fitted distribution with
- `>print(loglik.norm(MLest.norm,heri.serie))`

Focusing on Heritability

- The solution is given in the Analysis.r script file

- **Second Key Result**: Raw outputs are ambiguous, always confirm with a summary statistic and a test!

Evolution over multiple generations

Let's set the `>LociNum=10`

`>GenerationNum=12`

Does everybody see the trait evolving?

Evolution over multiple generations

Let's set the `>LociNum=10`

`>GenerationNum=12`

Does everybody see the trait evolving?

- With different demographic parameters
  - `>GenoNum=10`
  - `>GenoNum=3`
  - `>GenoNum=50`

Evolution over multiple generations

Let's set the `>LociNum=10`

`>GenerationNum=12`

Trait mean and variance is stable, heritability estimates swing a bit but nothing much is going on.

However, the frequency of the genotypes (originally 10 of each) has changed.

This is called drift within finite population

Now let's create mutants by setting `>MuRate=0.01`

Evolution over multiple generations

Let's set the `>LociNum=10`

`>GenerationNum=12`

Trait mean and variance is stable, heritability estimates swing a bit but nothing much is going on.

However, the frequency of the genotypes (originally 10 of each) has changed.

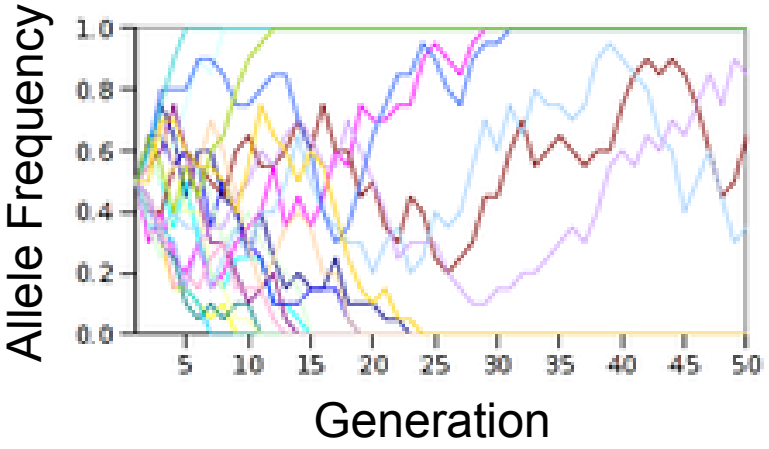
This is called drift within finite population

Now let's create mutants by setting `>MuRate=0.01`

Mutants have raised in frequency/invaded

Evolution over multiple generations

Key result: Mutation/Drift are the two stochastic processes triggering the fate of genetic diversity in natural population.



Reset `>Mu.Rate=0`

Now, we are going to explore the consequence of a parametric process affecting genetic diversity: **Selection!!!**

Selection over multiple generations

Let's select the bottom 50% of the population at every generation

```
>Sel.max=.5
```

What happens in terms of Genotype_Frequency?

How is the trait generally evolving (linearly or log-linearly)?

Selection over multiple generations

Let's select the bottom 50% of the population at every generation

`>Sel.max=.5`

What happens in terms of Genotype_Frequency?

How is the trait generally evolving (linearly or log-linearly)?

Let's introduce a fair bit of mutation with `>MuRate=0.01`

How did the response evolved?

Selection over multiple generations

Let's select the bottom 50% of the population at every generation

```
>Sel.max=.5
```

What happens in terms of Genotype_Frequency?

How is the trait generally evolving (linearly or log-linearly)?

Let's introduce a fair bit of mutation with `>MuRate=0.01`

How did the response evolved?

The response is more linear, mutation providing the raw material for selection to act upon.

Selection over multiple generations

Let's now see how the population can evolve if we gradually increase selection by `>EnvChangeRate=-0.2` with `>MuRate=0`

What happens?

Selection over multiple generations

Let's now see how the population can evolve if we gradually increase selection by `>EnvChangeRate=-0.2` with `>MuRate=0`

What happens?

The population goes extinct, it's hopeless!!!

How can we restore the hope?

Selection over multiple generations

Under `>GenoNum=10`

```
>GenoRep=10
>LociNum=10
>LociFX="Equal Loci"
>sdGeneticFX=1
>sdEnvironmentalFX=1
>GenerationNum=12
>Fraction=TRUE
>Sel.max=.5
>Sel.min=-Inf
>EnvChangeRate=-0.2
```

What is the mimimum >`MuRate` required to make sure that the population will survive 12 generations in 50% of the runs?

Selection over multiple generations

First, how to get the number of generation at which Selection.Simulator() stopped?

Selection over multiple generations

First, how to get the number of generation at which Selection.Simulator() stopped?

```
>ncol(Selection.Simulator()[[1]])
```

Second, you will need to get the 50%-tile of a statistical serie

```
>quantile(serie,.5)
```

And don't forget to set >`Graph=FALSE` :)

Selection over multiple generations

First, how to get the number of generation at which Selection.Simulator() stopped?

```
>ncol(Selection.Simulator()[[1]])
```

Second, you may need to get the 50%-tile of a statistical serie

```
>quantile(serie,.5)
```

Or alternatively you can just get the number of runs that reached 12 generations

```
>length(serie[serie==12])
```

And don't forget to set >`Graph=FALSE` :)

Selection over multiple generations

For the solution, I ended up with something like that:

```
>GeneNum.list=list()

>for (Mu.Rate in c(.02,.03,.04)) {

>print("_____________")
>print(paste("Mutation Rate =",Mu.Rate))

>GeneNum.serie=vector()
>for (i in 1:1000)
GeneNum.serie=c(GeneNum.serie,ncol(Selection.Simulator(MuRate=
Mu.Rate)[[1]]))
>GeneNum.list[[paste(Mu.Rate)]]=length(GeneNum.serie[GeneNum.s
erie==12])/length(GeneNum.serie)
>}
```

Activities to follow up

- What is the probability of having an ancestral genotype in the population after 12 generations?

Tip: to test for the presence of a character string in an element, use the following regular expression:

```
>grep("Geno",names(Selection.Simulator()[[2]]))
```

- For a precise estimate of heritability, is it better to have many genotypes with low replication or many replication of a few genotypes?

- Finally can you draw the evolution of the frequency of each genotype over time?

Tip:line 165 `>GenoFrequency=table(Gname)` *could be moved to before the closing bracket in line 161 and* `GenoFrequency` *should be changed from a simple vector to a list that*