Deriva genética

Curso de Evolución 2020

18/09/2020

Deriva genética

Introducción

Ya utilizamos la distribución binomial para entender temas vinculados, tales como la probabilidad de observar *i* copias de un alelo A, dada una frecuencia real de A en la población *p* y el tamaño de la muestra *n*. Entendemos que la frecuencia observada en la muestra *i/n* puede diferir de *p* puesto que la primera es una propiedad de la muestra mientras que la segunda es una característica de la población.

La deriva genética es el cambio de frecuencias alélicas en la población a lo largo del tiempo debido al azar. En genética de poblaciones, existen varios modelos que especifican cómo se produce dicho cambio. Uno de los más usados es el modelo Wright-Fisher, desarrollado en forma más o menos paralela por Sewall Wright y Ronald Fisher, dos de los fundadores del campo. En dicho modelo, los alelos de una generación se obtienen por un muestreo al azar con reposición de los alelos de la generación precedente. Para un sistema diploide, la población tiene 2N alelos. Cada uno de los 2N alelos de la generación $t-1$ se tomaron al azar (imaginemos, para visualizarlo, uno a uno) de los 2N disponibles en la generación t_0. Cada uno de los 2N alelos de la generación parental en t_0 tiene igual probabilidad de ser muestreado al escoger uno de la generación $t-1$. Por lo tanto, la probabilidad de que un alelo sea de tipo A en *t* es igual a la frecuencia de A en *t*-1. El proceso se repite generación tras generación, de modo que

p_0 ==> p_1 ==> $p_2 \ldots$

Antes de abordar estas simulaciones, recordemos qué sucede en cada generación, aplicando la distribución binomial.

Consideramos una población de *N* individuos. Nos interesa solamente saber que, para un sistema diploide, el número de alelos es *2N*, y necesitamos saber la frecuencia *p* del alelo A en la población inicial. Usando la función *dbinom*, calculamos la probabilidad de observar 0, 1, … 2N copias de A en la muestra. Como estamos evaluando todos los resultados posibles, verificamos que la suma de los *P(i)* es 1.

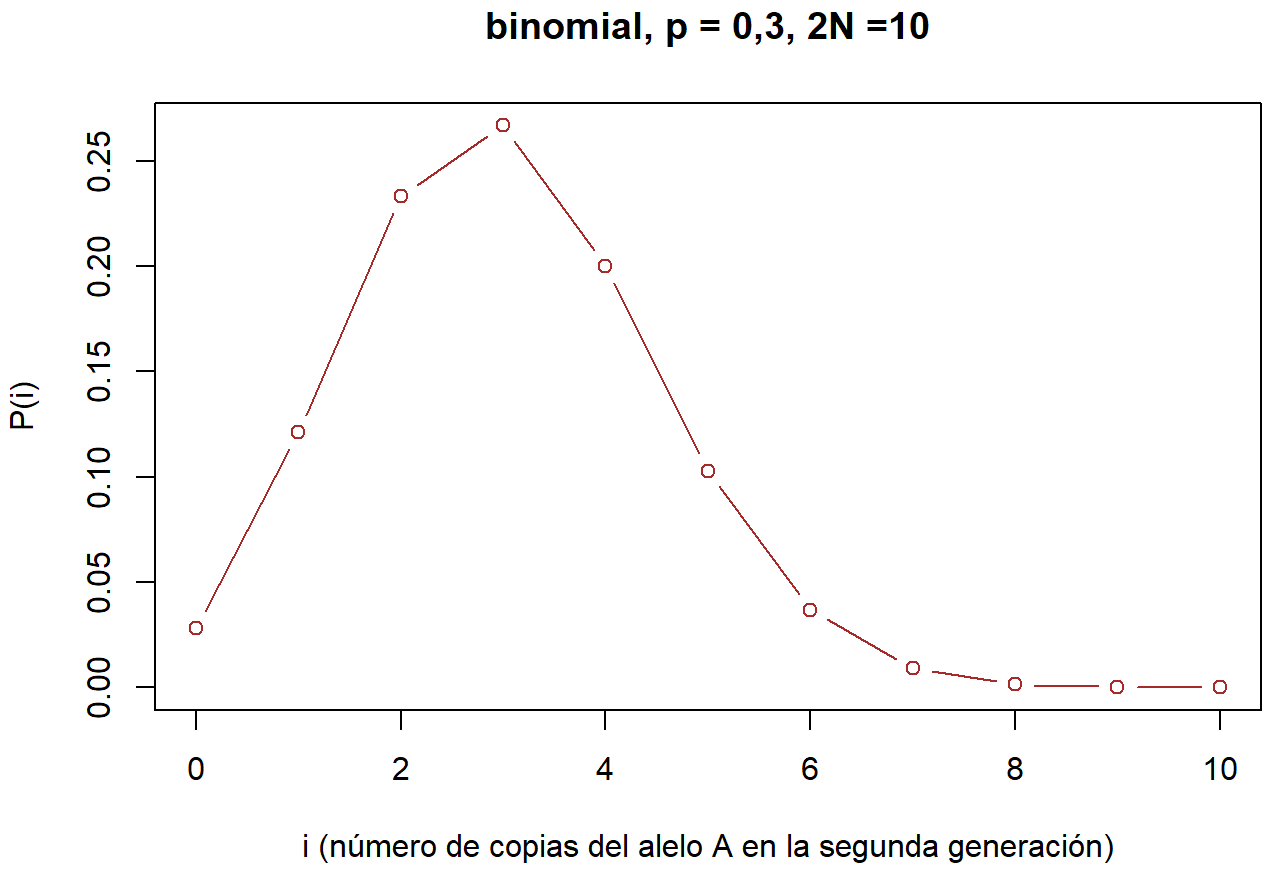
```
N  = 5  # número de individuos (diploides) en la población (asumimos que N es constante).
pr = 0.3 # valor inicial de p (frecuencia del alelo A) en la población

#A. Probabilidad de observar 0, 1, ... 2N copias del alelo A en una muestra de n=10, dado que
la frecuencia del
#   alelo en la población es pr

  dist = dbinom(c(0:(2*N)), 2*N, pr) # c(0:2*N) es el rango de resultados cuyas probabilidade
s queremos calcular
  print(dist)
```

```
##  [1] 0.0282475249 0.1210608210 0.2334744405 0.2668279320 0.2001209490
##  [6] 0.1029193452 0.0367569090 0.0090016920 0.0014467005 0.0001377810
## [11] 0.0000059049
```

```
  sum(dist) # como calculamos para todos los resultados posibles, verificamos que la suma da
 1
```

```
## [1] 1
```

```
#B. Graficamos los valores obtenidos
  plot(c(0:(2*N)), dist, type = "b",
       main = "binomial, p = 0,3, 2N =10",
       xlab = "i (número de copias del alelo A en la segunda generación)",
       ylab = "P(i)",
       col = "brown"
       )
```

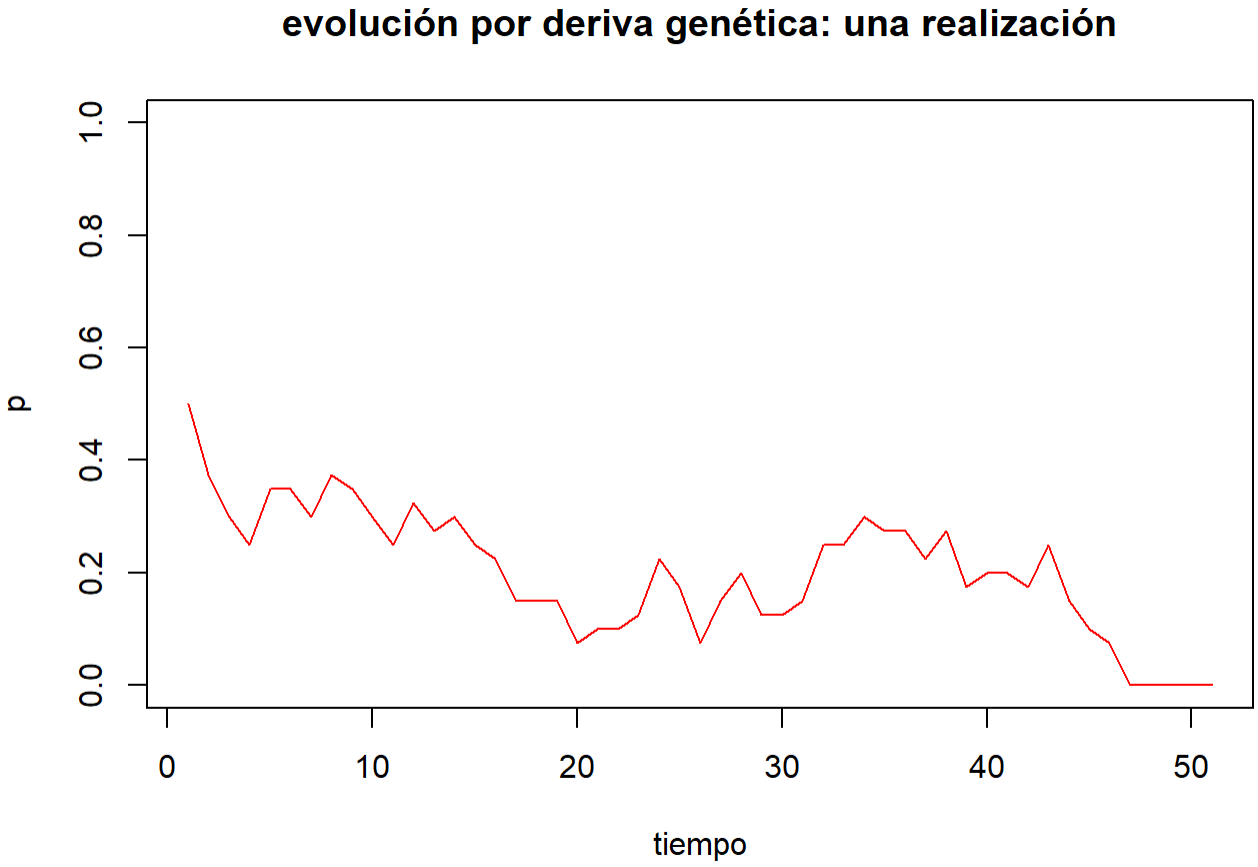


En esta aplicación de la binomial, en cada generación muestreamos todos los alelos de la población ($i=2N$) usando la frecuencia p de la generación inmediatamente precedente.

```r
# Condiciones de la simulación
  N  = 20 # número de individuos; como simulamos loci diploides, hay 2N alelos en la población
  p0 = 0.5 # frecuencia inicial de un alelo (A), cuya frecuencia en la población seguimos a lo largo del tiempo.
  t  = 50 # número de generaciones
  pvec = as.vector(p0) # creamos un vector que iremos "llenando" con las frecuencias alélicas de A para cada t

# Simulación: utilizamos la función "for" para realizar los muestreos basados en la distribución binomial de los 2N alelos partiendo desde p0 en el primer muestreo y luego tomando p1 como frecuencia inicial para el segundo muestreo (y así sucesivamente hasta el muestreo t=50 en este caso)
  p=p0 #frecuencia inicial
  for(i in seq(1:t)){
    p = rbinom(1, 2*N, p)/(2*N)
    # print(p)
    pvec = append(pvec, p) # el valor obtenido de p se agrega al final del vector de frecuencias pvec
  }
# Frecuencias inicial y final
  print(c(p0, pvec[length(pvec)]))
```

```
## [1] 0.5 0.0
```

```r
# Gráfica de la trayectoria de evolución por deriva genética de la simulación precedente
  plot(pvec, type = "l" ,main = "evolución por deriva genética: una realización", xlab = "tiempo", ylab = "p", ylim = range(0,1), col = "red")
```



Podemos usar la simulación de más arriba para ganar cierta intuición sobre las características de la deriva genética. Por ejemplo, podemos probar distintos tamaños poblacionales (modificando el valor de N) para ver cómo cambia el proceso. Del mismo modo, podemos experimentar con diferentes frecuencias iniciales o cambiar el número de generaciones.

Sigue un borrador para comparar dos procesos de deriva con el mismo punto de partida e idénticas condiciones generales.

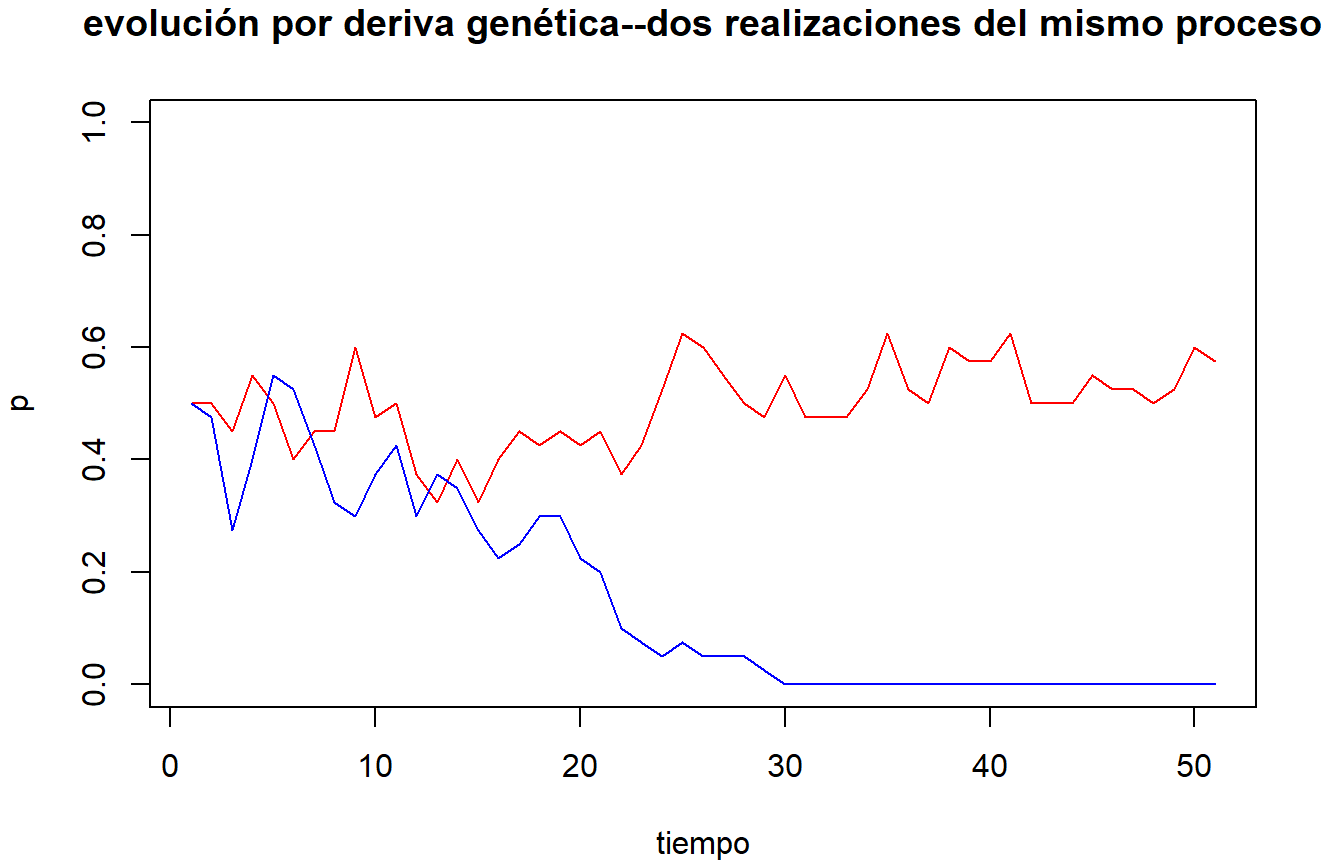
```
# Condiciones de la simulación
  # Nota: tomamos los valores de N, p0 y t del bloque anterior
  # En cambio, creamos dos vectores para registrar las frecuencias a lo largo del tiempo, com
enzando por asignarle a cada uno el valor de p0
    pvec1 = pvec2 = as.vector(p0) # inicializando dos vectores de frecuencias

# Simulación
  p1 = p2 = p0 # frecuencias iniciales

  for(i in seq(1:t)){
    p1 = rbinom(1, 2*N, p1)/(2*N) # muestreo de la binomial (en número de copias del alelo),
 divido por 2N para obener la frecuencia relativa
    pvec1 = append(pvec1, p1) # el valor obtenido de p1 se agrega al final del vector de frec
uencias

    p2 = rbinom(1, 2*N, p2)/(2*N) # lo mismo de arriba pero para la segunda simulación
    pvec2 = append(pvec2, p2)
  }

# Frecuencias inicial y final
  print(c("p(0) =", p0, "p(t) =", pvec1[length(pvec1)]))
```

```
## [1] "p(0) =" "0.5"    "p(t) =" "0.575"
```

```
  print(c("p(0) =", p0, "p(t) =", pvec2[length(pvec2)]))
```

```
## [1] "p(0) =" "0.5"    "p(t) =" "0"
```

```
# Gráfica de la trayectoria de evolución por deriva genética de la simulación precedente
  plot(pvec1, type = "l" ,main = "evolución por deriva genética--dos realizaciones del mismo
 proceso", xlab = "tiempo", ylab = "p", ylim = range(0,1), col = "red")
  lines(pvec2, col = "blue")
```



Comentarios

Las ideas principales que queremos reforzar simulando dos realizaciones (y repitiendo este experimento varias veces) del mismo proceso son:

- Concepto de proceso aleatorio: el proceso tiene reglas probabilísticas, no deterministas, por lo que dos procesos con las mismas reglas y el mismo punto de partida tienen diferentes trayectorias. Se dice que son dos realizaciones de un mismo proceso.
- Proceso markoviano, sin memoria: en cada paso, el proceso depende únicamente de su estado (en nuestro caso, la frecuencia del alelo de referencia) y de las reglas para pasar de dicho estado al siguiente.
- Estados absorbentes: una vez que se llega a la fijación (p=1) o eliminación (p=0) del alelo, no hay más cambios; esos dos estados son absorbentes (se puede llegar a ellos, pero no se puede salir de ellos).

Notemos, de paso, que las trayectorias “azul” y “roja” son independientes, aunque les marcamos un mismo punto de partida y siguen las mismas reglas (muestreo binomial con reposición, idéntico tamaño poblacional). Estas dos trayectorias pueden pensarse como:

- Ejemplos de dos posibles trayectorias de la frecuencia de un mismo alelo, partiendo de p0. Si la línea azul representa la trayectoria de un alelo en una población real, podemos pensar en la línea roja como otra trayectoria igualmente probable… y podríamos seguir agregando más y más trayectorias.
- La historia de dos genes no ligados (esto es, con trayectorias independientes) en una misma población, con idénticas frecuencias iniciales.

Efecto del tamaño poblacional

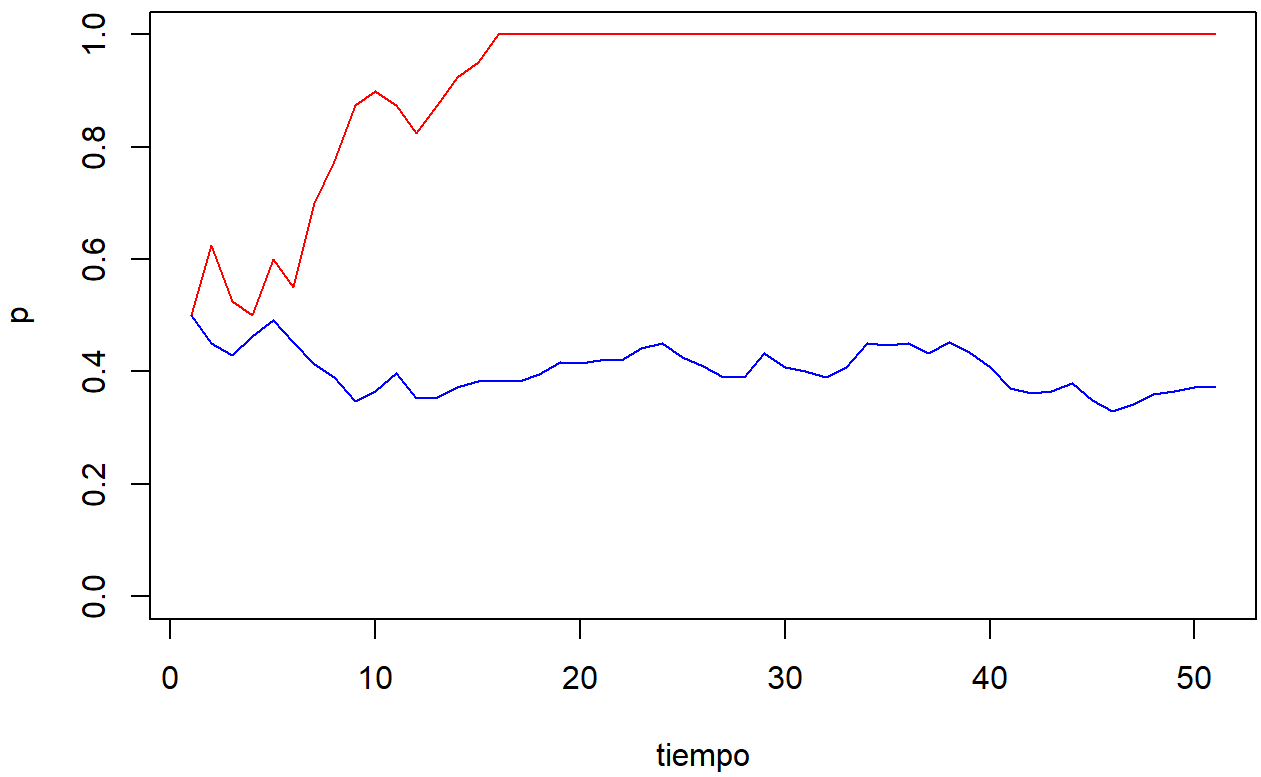
```
# Condiciones de la simulación
  # Nota: tomamos los valores de p0 y t de los bloques previos

# Pero usaremos dos tamaños poblacionales diferentes:
N1 = 20
N2 = 200


  # En cambio, creamos dos vectores para registrar las frecuencias a lo largo del tiempo, com
enzando por asignarle a cada uno el valor de p0
    pvec1 = pvec2 = as.vector(p0) # inicializando dos vectores de frecuencias

# Simulación
  p1 = p2 = p0 # frecuencias iniciales

  for(i in seq(1:t)){
    p1 = rbinom(1, 2*N1, p1)/(2*N1) # muestreo de la binomial (en número de copias del alel
o), divido por 2N para obener la frecuencia relativa
    pvec1 = append(pvec1, p1) # el valor obtenido de p1 se agrega al final del vector de frec
uencias

    p2 = rbinom(1, 2*N2, p2)/(2*N2) # lo mismo de arriba pero para la segunda simulación
    pvec2 = append(pvec2, p2)
  }

# Frecuencias inicial y final
  print(c("p(0) =", p0, "p(t) =", pvec1[length(pvec1)]))
```

```
## [1] "p(0) =" "0.5"    "p(t) =" "1"
```

```
  print(c("p(0) =", p0, "p(t) =", pvec2[length(pvec2)]))
```

```
## [1] "p(0) =" "0.5"    "p(t) =" "0.375"
```

```
# Gráfica de la trayectoria de evolución por deriva genética de la simulación precedente
  plot(pvec1, type = "l" ,
       main = "rojo: N1; azul: N2",
       xlab = "tiempo", ylab = "p",
       ylim = range(0,1), col = "red")
  lines(pvec2, col = "blue")
```


rojo: N1; azul: N2
p
0.0
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40
50
tiempo