Evolution

```
evol = function (mod, gen, f, pop, N, wAA, wAa, waa, p = FALSE) #1 Cria
funcao "evol" com os argumentos "mod", "gen", "f", "pop", "N", "wAA", "wAa",
"waa" e "p".
{
  #FUNCAO DE DERIVA GENETICA
  if (mod == "d") #2 Se o argumento "mod" for igual a "d", roda o modelo de
deriva genetica.
  {
    #VERIFICANDO OS PARAMETROS
    if (N != round (N) | N <= 0) #3 Verifica se "N" eh um numero inteiro e
maior que zero.
    {
      stop ("N deve ser um numero inteiro e > 0.") #4 Senao, interrompe a
funcao e exibe mensagem para o usuario.
    }
    if (gen != round (gen) | gen <= 0) #5 Verifica se "gen" eh um numero
inteiro e maior que zero.
    {
      stop ("gen deve ser um numero inteiro e > 0.") #6 Senao, interrompe a
funcao e exibe mensagem para o usuario.
    }
    if (f < 0 | f > 1 ) #7 Verifica se "f" esta no intervalo adequado para a
funcao.
    {
      stop ("f deve estar no intervalo 0 <= f <= 1.") #8 Senao, interrompe a
funcao e exibe mensagem para o usuario.
    }
    if (pop != round (pop) | pop <= 0) #5 Verifica se "pop" eh um numero
inteiro e maior que zero.
    {
      stop ("pop deve ser um numero inteiro e > 0.") #10 Senao, interrompe a
funcao e exibe mensagem para o usuario.
    }
    if (p != FALSE) #11 Verifica se o usuario inseriu algum valor para "p".
    {
      if (p < 0 | p > 1 ) #12 Se sim, verifica se "p" esta no intervalo
adequado para a funcao.
      {
        stop ("p deve estar no intervalo 0 <= p <= 1.") #13 Senao,
interrompe a funcao e exibe mensagem para o usuario.
      }
    }
    #CRIANDO OS OBJETOS DA FUNCAO
    simulacao = rep (NA, N) #14 Cria vetor "simulacao" com "N" NAs para
guardar os resultados da simulacao.
    populacao = c ((rep ("A", f * N)), rep ("a", N - (f * N))) #15 Cria
vetor "populacao" com as frequencias alelicas estabelecidas por "f".
```

```
    final = rep (NA, pop) #16 Cria vetor "final" com "pop" NAs para guardar
as frequencias alelicas finais de cada populacao.
    #PREPARANDO A AREA DOS GRAFICOS
    x11 () #17 Abre dispositivo de tela.
    par (mfrow = c (1, 2)) #18 Divide o dispositivo de tela em duas colunas.
    plot (x = NULL, y = NULL, #19 Plota area do grafico vazia.
          xlim = c (0, gen), ylim = c (0, 1), #20 Determina valores minimos
e maximos dos eixos X e Y.
          bty = "l", #21 Coloca margens nos lados 1 e 2.
          xlab = "Geracoes", ylab = "f(A)") #22 Coloca os titulos dos eixos
do grafico.
    #SIMULANDO AS TRAJETORIAS DAS POPULACOES
    for (j in 1:pop) #23 Entra em um fluxo "for" com contador "j" de 1 ate
"pop".
    {
      points (0, f, col = j, pch = 16, cex = 0.5) #24 Plota a primeira
frequencia alelica no grafico.
      f.old = f #25 Guarda o valor de "f" em "f.old".
      for (i in 1:gen) #26 Entra em um fluxo "for" com contador "i" de 1 ate
"gen".
      {
        simulacao = sample (populacao, replace = TRUE) #27 Amostra com
reposicao os alelos de "populacao" e guarda em "simulacao".
        f.new = (length (simulacao [simulacao == "A"])) / N #28 Calcula em
"f.new" a nova frequencia alelica.
        points (i, f.new, col = j, pch = 16, cex = 0.5) #29 Plota a nova
frequencia alelica no grafico.
        segments (i - 1, f.old, i, f.new, col = j) #30 Une os pontos das
frequencias alelicas.
        f.old = f.new #31 A nova frequencia alelica passa a ser a antiga
frequencia alelica para o proximo sorteio.
        populacao = simulacao #32 A populacao amostrada ("simulacao") passa
a ser a "populacao" para o proximo sorteio.
      }
      populacao = c ((rep ("A", f * N)), rep ("a", N - (f * N))) #33 Retorna
"populacao" para a frequencia alelica inicial.
      final [j] = f.new #34 Guarda a frequencia alelica final na posicao "j"
de "final".
    }
    #CRIANDO O HISTOGRAMA DAS FREQUENCIAS ALELILCAS FINAIS
    hist (final, xlab = "f(A) final", ylab = "Frequencia", main = NULL) #35
Plota histograma com as frequencias alelicas finais.
    #CALCULANDO O VALOR DE P
    if (p != FALSE) #36 Verifica se o usuario inseriu algum valor para "p".
    {
      p = round (p, digits = 1) #37 Se sim, estabelece o valor de "p" com
uma casa decimal.
      abline (v = p, col = "red") #38 Coloca linha vertical no histograma
com o valor de "p".
      mtext ("p", side = 3, at = p, col = "red") #39 Escreve "p" sobre a
```

```
linha criada em #38.
      final = c (round (final, digits = 1)) #40 Arredonda todos os elementos
de "final" para uma casa decimal.
      prob = (length (final [final == p])) / pop #41 Calcula a frequencia
que o valor de "p" aparece em "final".
      cat (paste ("Probabilidade de f(A) =", p, "em", pop, "populacoes com
tamanho", N, "apos", gen, "geracoes:", prob)) #42 Imprime mensagem na tela
com o valor de "p".
    }
  }
  #FUNCAO DE SELECAO NATURAL
  if (mod == "s") #43 Se o argumento "mod" for igual a "s", roda o modelo de
selecao natural.
  {
    #VERIFICANDO OS ARGUMENTOS
    if (any (f < 0 | f > 1)) #44 Verifica se os elementos de "f" estao no
intervalo adequado para a funcao.
    {
      stop ("f deve ser um vetor com elementos no intervalo 0 <= f <= 1.")
#45 Senao, interrompe a funcao e exibe mensagem para o usuario.
    }
    if (gen != round (gen) | gen <= 0) #46 Verifica se "gen" eh um numero
inteiro e maior que zero.
    {
      stop ("gen deve ser um numero inteiro e > 0.") #47 Senao, interrompe a
funcao e exibe mensagem para o usuario.
    }
    if (pop != round (pop) | pop <= 0) #48 Verifica se "pop" eh um numero
inteiro e maior que zero.
    {
      stop ("pop deve ser um numero inteiro e > 0.") #49 Senao, interrompe a
funcao e exibe mensagem para o usuario.
    }
    if (wAA < 0 | wAA > 1 ) #50 Verifica se "wAA" esta no intervalo adequado
para a funcao.
    {
      stop ("wAA deve estar no intervalo 0 <= wAA <= 1.") #51 Senao,
interrompe a funcao e exibe mensagem para o usuario.
    }
    if (wAa < 0 | wAa > 1 ) #52 Verifica se "wAa" esta no intervalo adequado
para a funcao.
    {
      stop ("wAa deve estar no intervalo 0 <= wAa <= 1.") #53 Senao,
interrompe a funcao e exibe mensagem para o usuario.
    }
    if (waa < 0 | waa > 1 ) #54 Verifica se "waa" esta no intervalo adequado
para a funcao.
    {
      stop ("waa deve estar no intervalo 0 <= waa <= 1.") #55 Senao,
interrompe a funcao e exibe mensagem para o usuario.
    }
```

```
    if (p != FALSE) #56 Verifica se o usuario inseriu algum valor para "p".
    {
      if (p < 0 | p > 1 ) #57 Se sim, verifica se "p" esta no intervalo
adequado para a funcao.
      {
        stop ("p deve estar no intervalo 0 <= p <= 1.") #58 Senao,
interrompe a funcao e exibe mensagem para o usuario.
      }
    }
    #CRIANDO OS OBJETOS DA FUNCAO
    semi.final = rep (NA, length (f)) #59 Cria vetor "semi.final" com o
comprimento de "f" preenchido com NAs para gaurdar as frequencias alelicas
finais.
    #PREPARANDO A AREA DOS GRAFICOS
    x11() #60 Abre dispositivo de tela.
    par (mfrow = c (1, 2)) #61 Divide o dispositivo de tela em duas colunas.
    plot (x = NULL, y = NULL, #62 Plota area do grafico vazia.
          xlim = c (0, gen), ylim = c (0, 1), #63 Determina valores minimos
e maximos dos eixos X e Y.
          bty = "l", #64 Coloca margens nos lados 1 e 2.
          xlab = "Geracoes", ylab = "f(A)") #65 Coloca os titulos dos eixos
do grafico.
    #SIMULANDO AS TRAJETORIAS DAS POPULACOES
    for (j in 1:length (f)) #66 Entra em um fluxo "for" com contador "j" de
1 ate o comprimento de "f".
    {
      points (0, f[j], pch = 16, cex = 0.5) #67 Plota a frequencia alelica
inicial no grafico.
      f.old = f[j] #68 Guarda o valor da frequencia alelica inicial em
"f.old".
      for (i in 1:gen) #69 Entra em um fluxo "for" com contador "i" de 1 ate
"gen".
      {
        f.new = (((f.old * f.old) * wAA) + (f.old * (1 - f.old) * wAa)) /
#70 Calcula em "f.new" a nova frequencia alelica.
                (((f.old * f.old) * wAA) + (2 * f.old * (1 - f.old) * wAa) +
(((1 - f.old) * (1 - f.old)) * waa))
        points (i, f.new, col = j, pch = 16, cex = 0.5) #71 Plota a nova
frequencia alelica no grafico.
        segments (i - 1, f.old, i, f.new, col = j) #72 Une os pontos das
frequencias alelicas.
        f.old = f.new #73 A nova frequencia alelica passa a ser a antiga
frequencia alelica para o proximo sorteio.
      }
      semi.final [j] = f.new #74 Guarda a frequencia alelica final na
posicao "j" de "semi.final".
    }
    #CRIANDO O HISTOGRAMA DAS FREQUENCIAS ALELILCAS FINAIS
    final = rep (semi.final, each = pop) #75 Como o modelo eh
deterministico, nao ha necessidade de repetir as simulacoes para as mesmas
```

```
frequencias alelicas iniciais.
                                          #Assim, no vetor "final" os valores
das simulacoes sao repetidos para as simulacoes com as mesmas frequencias
iniciais.
    hist (final, xlab = "f(A) final", ylab = "Frequecia", main = NULL) #76
Plota histograma com as frequencias alelicas finais.
    #CALCULANDO O VALOR DE P
    if (p != FALSE) #77 Verifica se o usuario inseriu algum valor para "p".
    {
      p = round (p, digits = 1) #78 Se sim, estabelece o valor de "p" com
uma casa decimal.
      abline (v = p, col = "red") #79 Coloca linha vertical no histograma
com o valor de "p".
      mtext ("p", side = 3, at = p, col = "red")  #80 Escreve "p" sobre a
linha criada em #79.
      final = c (round (final, digits = 1)) #81 Arredonda todos os elementos
de "final" para uma casa decimal.
      prob = round ((length (final [final == p])) / (pop * length (f)),
digits = 2) #82 Calcula a frequencia que o valor de "p" aparece em "final".
      cat (paste ("Probabilidade de f(A) =", p, "após", gen, "geracoes:",
prob)) #83 Imprime mensagem na tela com o valor de "p".
    }
  }
}
```

From:
http://ecor.ib.usp.br/ - **ecoR**

Permanent link:
http://ecor.ib.usp.br/doku.php?id=05_curso_antigo:r2017:alunos:trabalho_final:carol.mendonca.bio:evol

Last update: **2020/08/12 06:04**