Simulation on Calafell et al, 1999

Bo Peng (bpeng@rice.edu)

28th February 2005

1 Introduction

This paper

> F. Calafell, EL Grigorenko, AA Chikanian and KK Kidd, Haplotype Evolution and Linkage Disequilibrium: A simulation Study (2001) Human Heridity, 51:85-96

simulated the evolution of a three-site haplotype system, two restriction fragment length polymorphisms flanking one short tandem repeart polymorphism, under five different demographic scenarios.

The simulation was done through a specialized program which is not available now. The following repeats this analysis using simuPOP.

2 Model

2.1 Genotypic structure

- B1 biallelic RFLP, two alleles B1 B2
- (CA)n STRP, seven alleles CA11 - CA17 (1-7 in simuPOP)
- A1 biallelic RFLP, two alleles A1 A2

2.2 Recombination

- B1 - CA: 0.006%
- CA - A1: 0.020%

simuPOP code

```
recombine = recombinator(   rates = [0.00006, 0.0002],   afterLoci=[0,1] )
```

2.3 Mutation

- CA symmetric stepwise mutation model
  - mutation exceeds boundaries are discaded. (e.g. 11->10, 17->18))
  - rate of mutate one: 1e-4
  - mutate two steps: 2e-5
  - mutate three steps: 4e-6
  - ... at 1/5 of previous rate
  - Average mutation rate: 2.47e-4 (?)
  - Average change number: 1.713 (?)
- A1 10^{-8}
- B1 10^{-8}

simuPOP code:

```
# with r = 1e-4, call this func
def step():
  return int(-math.log(random.uniform(0,1))/math.log(5))+1
# mutators
mutate02 = kamMutator( atLoci=[0,2], rate=1e-8, maxAllele=2)
mutate1 =  gsmMutator( atLoci=[1], maxAllele=7, rate=1e-4, func = step)
```

2.4 Initial population

Two haplotypes:

- b1-ca13-a1 40%
- b2-ca14-a2 60%

simuPOP code

```
init = initByValue(   values = [ [ 1, 3, 1], [2, 4, 2] ],
proportions = [.4, .6 ]  )
```

2.5 Statistical measurements

D' , allele frequency and percentage of fixation.

- model 1,2,3 at 1000, 2500 and 5000 generation
- model 4, 5 at 5000 generation
- D'

  - between A and B
  - between A and CA11-CA17
  - between B and CA11-CA17

simuPOP code: (calculate statistics every 10 generations)

```
LD_Freq = stat(
  LD= [[0,2]] + [[0,1,1,x] for x in range(0,8)]
    + [[1,2,x,1] for x in range(0,8)],
  alleleFreq=[0,1,2],
  step=10)
```

2.6 Demographic models

Five of them

- 2N=2000, 5000 gen
- 2N=4000, 5000 gen
- 2N=10000,5000 gen
- 2N=2848 for 2786 gen, 2N=10,000 aferwards
- 2N=3600 for 3400 gen, reaching 2N=10,000 at 5000 gen

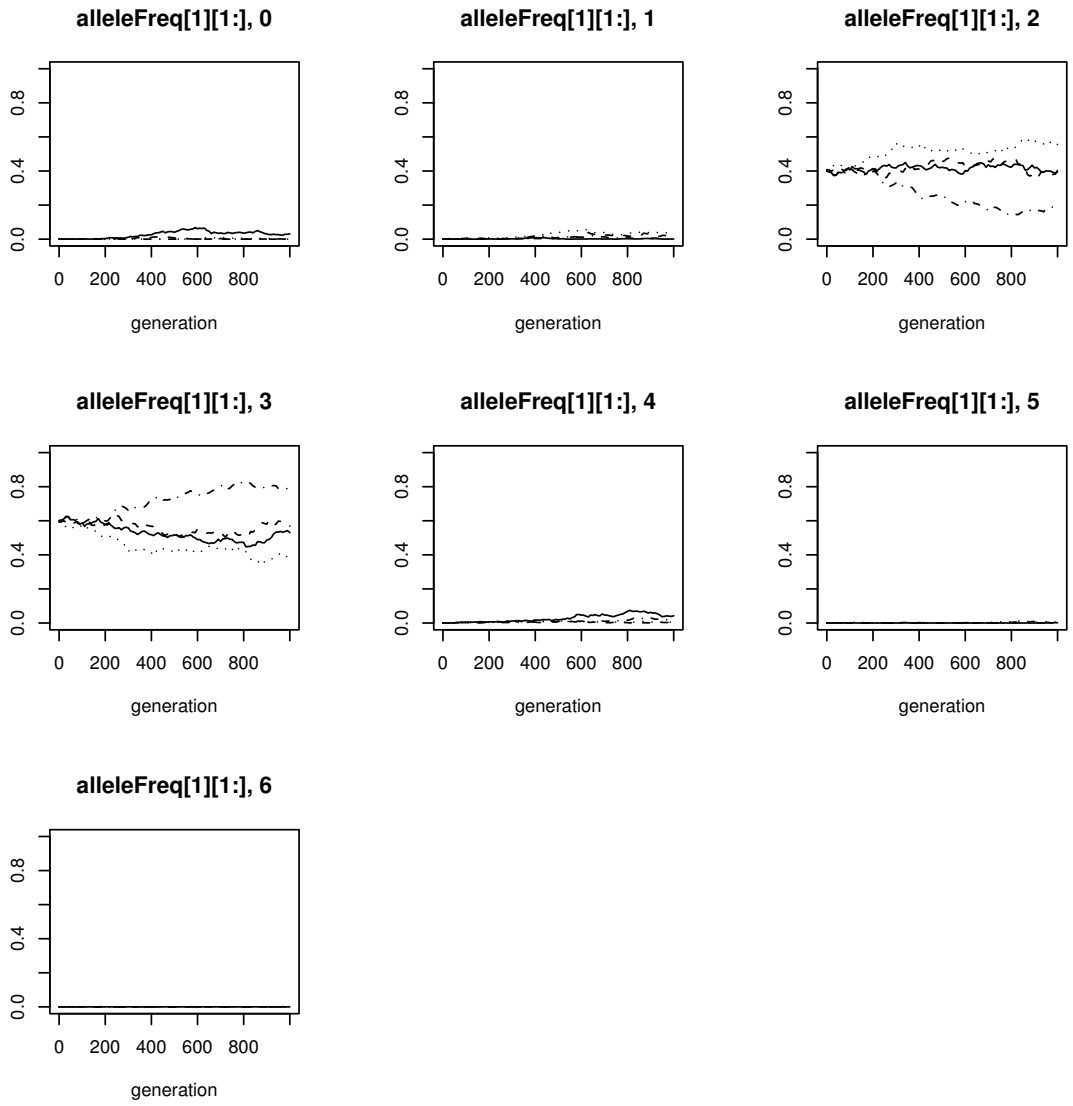
simuPOP code:

```
def scenario_1(gen):
  return 2000
def scenario_2(gen):
  return 5000
def scenario_3(gen):
  return 10000
def scenario_4(gen):
  if gen < 2786:
    return 2848
  elif gen < 3000: # rapid linear growth
    return (10000-2848)/(3000-2786)*(gen-2786)+2848
  else:
    return 10000
def scenario_5(gen):
  if gen < 3400:
    return 3600
  else:
    return (10000-3600)/(5000-3400)*(gen-3400)+3600
```

3 Results

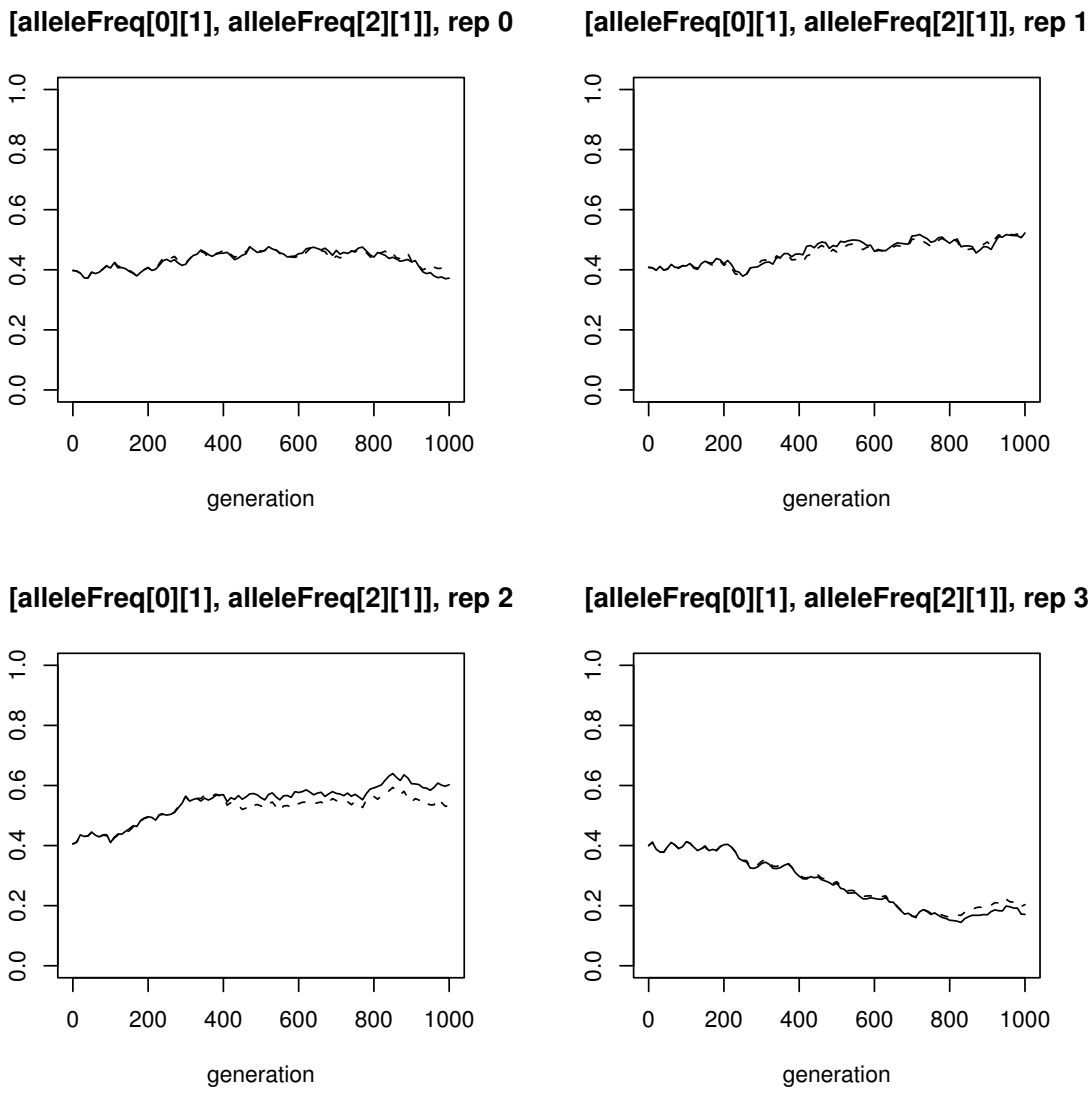
3.1 Allele frequency for seven CA alleles. (4 replicates)

1000 generations. Four duplicates at each subplot. Initially there are only allele 11 and 13 (subplot 2 and 3).



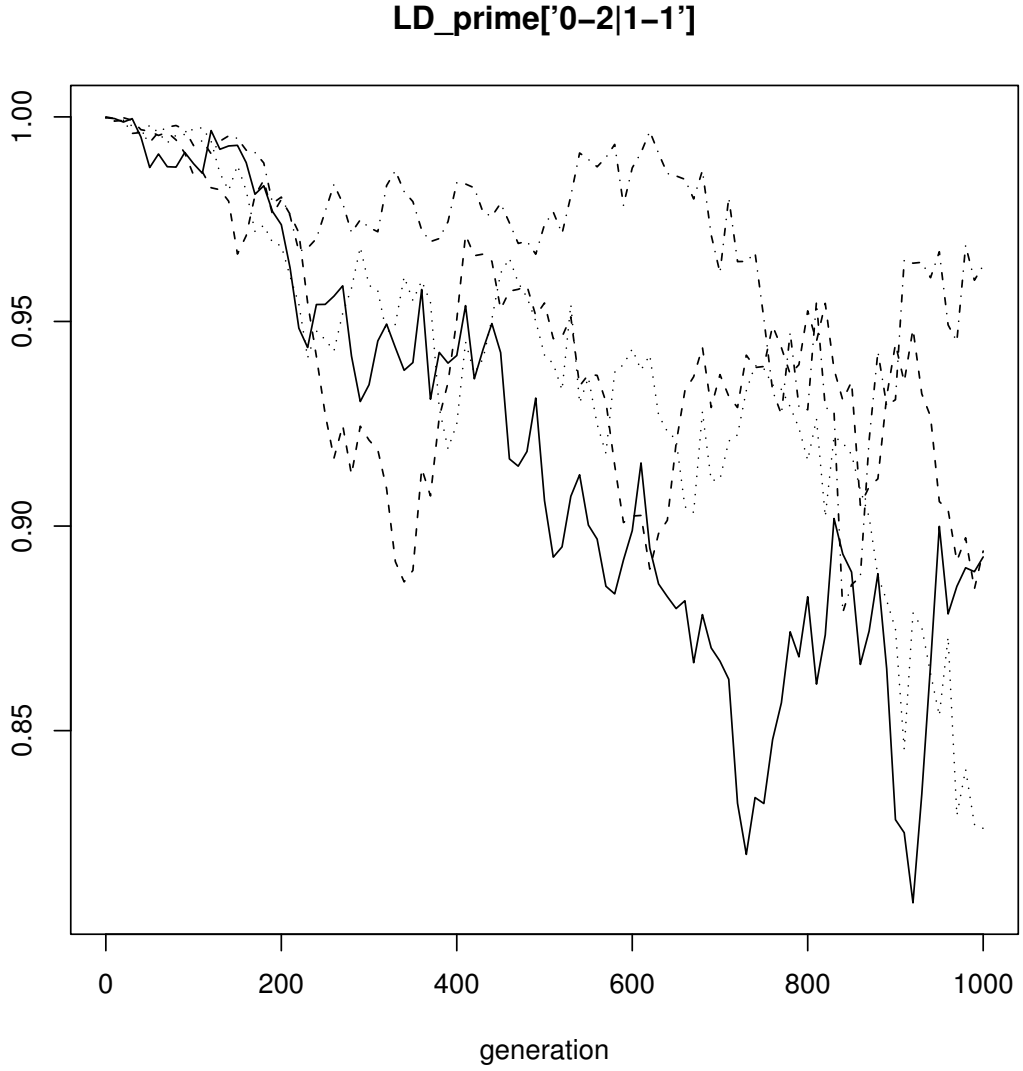
3.2 Allele frequency for allele 1's at loci A1 and B1:

Each subplot correspinds to a replicate.



3.3 LD' between A1 and B1

Four replicates. D' between Al and B1 loci (both are biallelic).



3.4 Allele frequency of allele CA11-CA17 at 1000 generation.

```
>>> listVars(analyze(simu, 100, 1000))
 std :
   [ 0 ]        0.011346778522
   [ 1 ]        0.0247507512181
   [ 2 ]        0.104112315881
   [ 3 ]        0.104139535647
   [ 4 ]        0.0232730214904
   [ 5 ]        0.0121670864498
   [ 6 ]        0.00379576754592
 mean :
   [ 0 ]        0.005731
   [ 1 ]        0.022131
   [ 2 ]        0.383415
   [ 3 ]        0.554762
   [ 4 ]        0.026734
   [ 5 ]        0.0060785
   [ 6 ]        0.0011485
>>> >>> listVars(analyze(simu, 100, 2500))
 std :
   [ 0 ]        0.0271180308088
   [ 1 ]        0.0595428383179
   [ 2 ]        0.15471827923
   [ 3 ]        0.147393343478
   [ 4 ]        0.0524258482203
   [ 5 ]        0.0239261372813
   [ 6 ]        0.00858153483658
 mean :
   [ 0 ]        0.017191
   [ 1 ]        0.050301
   [ 2 ]        0.3701775
   [ 3 ]        0.4886495
   [ 4 ]        0.055472
   [ 5 ]        0.0153055
   [ 6 ]        0.0029035
>>> listVars(analyze(simu, 100, 5000))
 std :
   [ 0 ]        0.0479677815273
   [ 1 ]        0.104127144013
   [ 2 ]        0.157893165937
   [ 3 ]        0.165757707119
   [ 4 ]        0.0930474046596
   [ 5 ]        0.0499564350614
   [ 6 ]        0.0219596916088
```

```
      mean :
        [ 0 ]        0.032004
        [ 1 ]        0.096951
        [ 2 ]        0.299037
        [ 3 ]        0.4348555
        [ 4 ]        0.1012575
        [ 5 ]        0.02709
        [ 6 ]        0.008805
```

4 Source code (with lots of comments)

```
#
# aimulation from F. Calcfell, EL Grigorenko, Kidd's paper
#
# Haplotype Evolution and Linkage Disequlibrium: A simulation
# study

from simuPOP  import *
from simuUtil import *
from simuRPy  import *
import math, random
#
# loci:
#   B1    biallelic RFLP    B1 B2
#   (CA)n STRP              CA11 - CA17
#   A1    biallelic RFLP:   A1 A2

#
# recombination
#   B1 - CA:  0.006%
#   CA - A1:  0.020%

recombine = recombinator(
  rates = [0.00006, 0.0002],
  afterLoci=[0,1] )

#
# Mutation:
#
#   CA  symmetric stepwise mutation model
#     mutation exceeds boundaries are discaded. (e.g. 11->10, 17->18))
#     rate of mutate one:  1e-4
#     mutate two steps: 2e-5
#     mutate three steps: 4e-6
#     ... at 1/5 of previous rate
```

```
#
#     (? Average mutation rate: 2.47e-4
#       Average change number: 1.713
#       Result of previous settings? ?)
#
#   A1 e-8
#   B1 e-8

# with r = 1e-4, call this func
def step():
  return int(-math.log(random.uniform(0,1))/math.log(5))+1

# test this step machanism:
# with 5 times less likely to go a step further
#s = [0] * 10000
#for i in range(0,10000):
#  s[i] = step()
## table:
#for i in range(0,10):
#  print s.count(i)

# mutators
mutate02 = kamMutator( atLoci=[0,2], rate=1e-8, maxAllele=2)
mutate1 =  gsmMutator( atLoci=[1], maxAllele=7, rate=1e-4, func = step)

#
# initial generation
#   b1-ca13-a1   40%
#   b2-ca14-a2   60%
#
init = initByValue(
  values = [ [ 1, 3, 1], [2, 4, 2] ],  # two genotypes
  proportions = [.4, .6 ]              # by proportion
)

#
# measurements:
#   model 1,2,3 at 1000, 2500 and 5000 gen
#   model 4, 5 at 5000 gen
#
# Meaaure: D' (-1 ~ 1 )
#   between A and B
#   between A and CA11-CA17
#   between B and CA11-CA17

LD_Freq = stat(
```

```
  LD= [[0,2]] + [[0,1,1,x] for x in range(0,8)]
    + [[1,2,x,1] for x in range(0,8)],
  alleleFreq=[0,1,2],
  step=10)   # calculate statistics every 10 steps

# demographics scenarios:
#
# 1. 2N=2000, 5000 gen
# 2. 2N=4000, 5000 gen
# 3. 2N=10000,5000 gen
# 4. 2N=2848 for 2786 gen, 2N=10,000 aferwards
# 5. 2N=3600 for 3400 gen, reaching 2N=10,000 at 5000 gen

# the following functions defines how population size
# will change with generation. The return values
# should be an array of subpopulation sizes. In this
# case, we should return an array of size one.
def scenario_1(gen):
  return [2000]

def scenario_2(gen):
  return [5000]

def scenario_3(gen):
  return [10000]

def scenario_4(gen):
  if gen < 2786:
    return 2848
  elif gen < 3000: # rapid linear growth
    return [(10000-2848)/(3000-2786)*(gen-2786)+2848]
  else:
    return [10000]

def scenario_5(gen):
  if gen < 3400:
    return [3600]
  else:
    return [(10000-3600)/(5000-3400)*(gen-3400)+3600]


########################

def simulation(nRep, scenario, endGen, visualizers=[]):

  # one chromosome with three loci
```

```
  pop = population( subPop=scenario(0), ploidy=2, loci=[3])

  # simulator with random mating
  simu = simulator(pop,
    randomMating(newSubPopSizeFunc=scenario),   # calulate new population size
    rep=nRep)
  # evolve
  simu.evolve(
    preOps = [ init ],
    ops = [
      LD_Freq,   # check LD' value and allele frequency
      recombine,
      mutate02,  # mutate loci 0 and 2
      mutate1,   # mutate locus 1
      # plot LD' between B1 and A1 loci
      varPlotter("LD_prime['0-2|1-1']", numRep=nRep,
        step=10, update=10, saveAs="LDprime"),
      # plot allele frequencies of B1 A1
      varPlotter("[alleleFreq[0][1], alleleFreq[2][1]]",
        numRep=nRep, byRep=True, step=10, update=10, varDim=2,
        ylim=[0,1],saveAs="alleleFreq02"),
      # plot allele frequencies at CA locus
      varPlotter("alleleFreq[1][1:]",
        numRep=nRep, byVal=True, step=10, update=10, varDim=7,
        ylim=[0,1], saveAs="AlleleFreq1"),
      # keep track of the following values at different
      # generation
      collector( expr='alleleFreq[1]', name = 'CA_af',
        at=[500,1000,1500,2000,2500,3000,3500,4000,4500,5000]),
      # report progress
      pyEval(r'"%d\n" % gen',rep=REP_LAST)
      ] + visualizers ,
    end = endGen,
    dryrun=False
  )
  return simu

# run!
#simu = simulation(100, scenario_3, 5000)

#simu.saveSimulator("sec3_99_5000.dat", format="bin")

import math

def analyze(simu, nrep, gen):
  d = []
```

```
  mean = [0]*7
  std = [0]*7
  for i in range(0, nrep):
    d.append(simu.dvars(i))
    # calulate mean frequency of CA 11 - 17
  for a in range(0,7):
    fq = [0]*nrep
    sum1 = 0.
    sum2 = 0.
    for i in range(0, 100):
      fq[i] = d[i].CA_af[gen][a+1]
      sum1 += fq[i]
      sum2 += fq[i]*fq[i]

    mean[a] = sum1/100.
    std[a] = math.sqrt((sum2 - nrep*mean[a]*mean[a])/(nrep-1))
  return {'mean':mean, 'std':std}

#listVars(analyze(simu, 100, 1000))
#listVars(analyze(simu, 100, 2500))
#listVars(analyze(simu, 100, 5000))

simu = simulation(4, scenario_3, 1000)
```