# Forward-time simulations using simuPOP, a tutorial

Bo Peng, Ph.D.

Department of Epidemiology
U.T. M.D. Anderson Cancer Center
Houston, TX

June. 6, 2007
Programmers' Cross Training
U.T. M.D. Anderson Cancer Center

# outline



1 What is simuPOP

2 An example

3 simuPOP components

# simuPOP is ...

simuPOP tutorial

Bo Peng, Ph.D.

What is simuPOP

An example

simuPOP components

**A forward-time population genetics simulation environment**

# simuPOP is ...



**A forward-time population genetics simulation environment**

- A population genetics simulation program

# simuPOP is ...



**A forward-time population genetics simulation environment**

- A population genetics simulation program
- Not coalescent-based

# simuPOP is ...



**A forward-time population genetics simulation environment**

- A population genetics simulation program
- Not coalescent-based
- Based on an object-oriented scripting language (Python)

# What simuPOP does

simuPOP tutorial

Bo Peng, Ph.D.

What is simuPOP

An example

simuPOP components

## simuPOP provides

- a large number of functions to manipulate populations, copy, split, merge, modify genotype, modify individuals, determine affection status, save to and load from various formats, generate sample, ...

# What simuPOP does

simuPOP tutorial

Bo Peng, Ph.D.

What is simuPOP

An example

simuPOP components

## simuPOP provides

- a large number of functions to manipulate populations, copy, split, merge, modify genotype, modify individuals, determine affection status, save to and load from various formats, generate sample, ...
- and a mechanism to evolve populations forward in time, subject to arbitrary demographic and genetic forces such as population size changes, mutation, migration, recombination, selection ...

# This is fun, but is it useful?



simuPOP can simulate the change of the genetic composition of a population in a complicated evolutionary process. It can be used to

- demonstrate population genetics phenomina

# This is fun, but is it useful?



simuPOP can simulate the change of the genetic composition of a population in a complicated evolutionary process. It can be used to

- demonstrate population genetics phenomina
- study the impact of genetic and demographic forces on the evolution of a population

# This is fun, but is it useful?



simuPOP can simulate the change of the genetic composition of a population in a complicated evolutionary process. It can be used to

- demonstrate population genetics phenomina
- study the impact of genetic and demographic forces on the evolution of a population
- study the evolution of (complex) genetic diseases

# This is fun, but is it useful?



simuPOP can simulate the change of the genetic composition of a population in a complicated evolutionary process. It can be used to

- demonstrate population genetics phenomina
- study the impact of genetic and demographic forces on the evolution of a population
- study the evolution of (complex) genetic diseases
- simulate samples to validate gene-mapping methods

# This is fun, but is it useful?



simuPOP can simulate the change of the genetic composition of a population in a complicated evolutionary process. It can be used to

- demonstrate population genetics phenomina
- study the impact of genetic and demographic forces on the evolution of a population
- study the evolution of (complex) genetic diseases
- simulate samples to validate gene-mapping methods
- study ascertainment methods in simulated populations

# This is fun, but is it useful?



simuPOP can simulate the change of the genetic composition of a population in a complicated evolutionary process. It can be used to

- demonstrate population genetics phenomina
- study the impact of genetic and demographic forces on the evolution of a population
- study the evolution of (complex) genetic diseases
- simulate samples to validate gene-mapping methods
- study ascertainment methods in simulated populations
- ...

# I heard about coalescent...



## Backward-time

- Start from a sample with unknown genotype
- Coalesce individuals until the most recent common ancestor of all individuals is found
- Starting from the MRCA, proceed forward in time and fill the genotype of each individual

# I heard about coalescent...

## Backward-time

- Start from a sample with unknown genotype
- Coalesce individuals until the most recent common ancestor of all individuals is found
- Starting from the MRCA, proceed forward in time and fill the genotype of each individual

## Forward-time

- Start from an initial population
- Evolve forward in time, generation by generation, subject to certain number of genetic and/or demographic effects
- Samples are collected from the last several generations

# Forward vs. backward-time simulations



## Backward-time

- Sample based, efficient.

## Forward-time

- Population based, inefficient.

# Forward vs. backward-time simulations

## Backward-time

- Sample based, efficient.
- Limited selection, recombination models and mating schemes

## Forward-time

- Population based, inefficient.
- Can simulate almost arbitrary evolutionary scenarios

# Forward vs. backward-time simulations

## Backward-time

- Sample based, efficient.
- Limited selection, recombination models and mating schemes
- Can not study population properties, or properties of ancestral generations

## Forward-time

- Population based, inefficient.
- Can simulate almost arbitrary evolutionary scenarios
- Can study population properties and ancestral generations

# Forward vs. backward-time simulations

## Backward-time

- Sample based, efficient.
- Limited selection, recombination models and mating schemes
- Can not study population properties, or properties of ancestral generations
- Used mostly for sample generation

## Forward-time

- Population based, inefficient.
- Can simulate almost arbitrary evolutionary scenarios
- Can study population properties and ancestral generations
- Wider application area

# On the simulations of complex human diseases



## Backward-time

- Haploid simulation only

## Forward-time

- No limit on ploidy

# On the simulations of complex human diseases



## Backward-time

- Haploid simulation only
- Additive selection and penetrance models

## Forward-time

- No limit on ploidy
- Arbitrary selection and penetrance models

# On the simulations of complex human diseases



## Backward-time

- Haploid simulation only
- Additive selection and penetrance models
- One disease susceptibility locus

## Forward-time

- No limit on ploidy
- Arbitrary selection and penetrance models
- Multiple DSL with interaction

# On the simulations of complex human diseases



## Backward-time

- Haploid simulation only
- Additive selection and penetrance models
- One disease susceptibility locus
- Generate independent samples

## Forward-time

- No limit on ploidy
- Arbitrary selection and penetrance models
- Multiple DSL with interaction
- Simulate populations, which allows more flexible sampling

# I like it, but, oohm, Python??

- For efficiency, the core of simuPOP is written in C++
- Python is used to
  - wrap simuPOP core (the glue language)
  - write simuPOP extensions (GUI etc)
  - pass parameters and more

# Availability



- simuPOP website: `http://simupop.sourceforge.net`
- Mailing list: `simupop-list@lists.sourceforge.net`
- License: GPL 2.0
- Platforms: `all OS on which Python is available`
- Monthly release, currently at 0.7.10
- Documentation: *simuPOP User's Guide* and *simuPOP Reference Manual*

# A simple example

```
>>> from simuPOP import *
>>> simu = simulator(
...     population(size=1000, ploidy=2, loci=[2]),
...     randomMating(),
...     rep = 3)
>>> simu.evolve(
...     preOps = [initByValue([1,2,2,1])],
...     ops = [
...         recombinator(rate=0.1),
...         stat(LD=[0,1]),
...         pyEval(r"'%3d   ' % gen", rep=0, step=10),
...         pyEval(r"'%f    ' % LD[0][1]", step=10),
...         pyEval(r"'\n'", rep=REP_LAST, step=10)
...     ],
...     end=100
... )
```

# Output of the example

```
  0    0.198036    0.200709    0.197748
 10    0.064744    0.063100    0.078473
 20    0.013233    0.019795    0.041057
 30    0.002985    0.004931    0.000649
 40    0.023492    0.002948    0.004462
 50    0.006016    0.014262    0.013900
 60    0.011310    0.008717    0.013715
 70    0.016652    0.014545    0.014426
 80    0.007250    0.002506    0.014372
 90    0.016994    0.014455    0.004147
100    0.000425    0.016570    0.008704
```

# simuPOP modules



```
>>> from simuPOP import *
>>> simu = simulator(
...     population(size=1000, ploidy=2, loci=[2]),
...     randomMating(),
...     rep = 3)
```

Import the default simuPOP module

# population

```
>>> from simuPOP import *
>>> simu = simulator(
...     population(size=1000, ploidy=2, loci=[2]),
...     randomMating(),
...     rep = 3)
```

Create a `population` of 1000 `diploid` individuals, each having two `loci` on the first chromosome

# simulator and mating scheme



```
>>> from simuPOP import *
>>> simu = simulator(
...     population(size=1000, ploidy=2, loci=[2]),
...     randomMating(),
...     rep = 3)
```

Create a `simulator` that has one replicate of this population, and a random mating scheme

# Operators!

```
>>> from simuPOP import *
>>> simu = simulator(
...     population(size=1000, ploidy=2, loci=[2]),
...     randomMating(),
...     rep = 3)
>>> simu.evolve(
...     preOps = [initByValue([1,2,2,1])],
...     ops = [
...         recombinator(rate=0.1),
...         stat(LD=[0,1]),
...         pyEval(r"'%3d   ' % gen", rep=0, step=10),
...         pyEval(r"'%f    ' % LD[0][1]", step=10),
...         pyEval(r"'\n'", rep=REP_LAST, step=10)
...     ],
...     end=100
... )
```

`initByValue` is applied before evolution

# Operators!

```
>>> from simuPOP import *
>>> simu = simulator(
...     population(size=1000, ploidy=2, loci=[2]),
...     randomMating(),
...     rep = 3)
>>> simu.evolve(
...     preOps = [initByValue([1,2,2,1])],
...     ops = [
...         recombinator(rate=0.1),
...         stat(LD=[0,1]),
...         pyEval(r"'%3d   ' % gen", rep=0, step=10),
...         pyEval(r"'%f    ' % LD[0][1]", step=10),
...         pyEval(r"'\n'", rep=REP_LAST, step=10)
...     ],
...     end=100
... )
```

`recombinator` is applied at every generation when an offspring is produced

# Operators!

```
>>> from simuPOP import *
>>> simu = simulator(
...     population(size=1000, ploidy=2, loci=[2]),
...     randomMating(),
...     rep = 3)
>>> simu.evolve(
...     preOps = [initByValue([1,2,2,1])],
...     ops = [
...         recombinator(rate=0.1),
...         stat(LD=[0,1]),
...         pyEval(r"'%3d   ' % gen", rep=0, step=10),
...         pyEval(r"'%f    ' % LD[0][1]", step=10),
...         pyEval(r"'\n'", rep=REP_LAST, step=10)
...     ],
...     end=100
... )
```

`stat` is applied to the offspring generation at every generation

# Operators!

```
>>> from simuPOP import *
>>> simu = simulator(
...     population(size=1000, ploidy=2, loci=[2]),
...     randomMating(),
...     rep = 3)
>>> simu.evolve(
...     preOps = [initByValue([1,2,2,1])],
...     ops = [
...         recombinator(rate=0.1),
...         stat(LD=[0,1]),
...         pyEval(r"'%3d   ' % gen", rep=0, step=10),
...         pyEval(r"'%f    ' % LD[0][1]", step=10),
...         pyEval(r"'\n'", rep=REP_LAST, step=10)
...     ],
...     end=100
... )
```

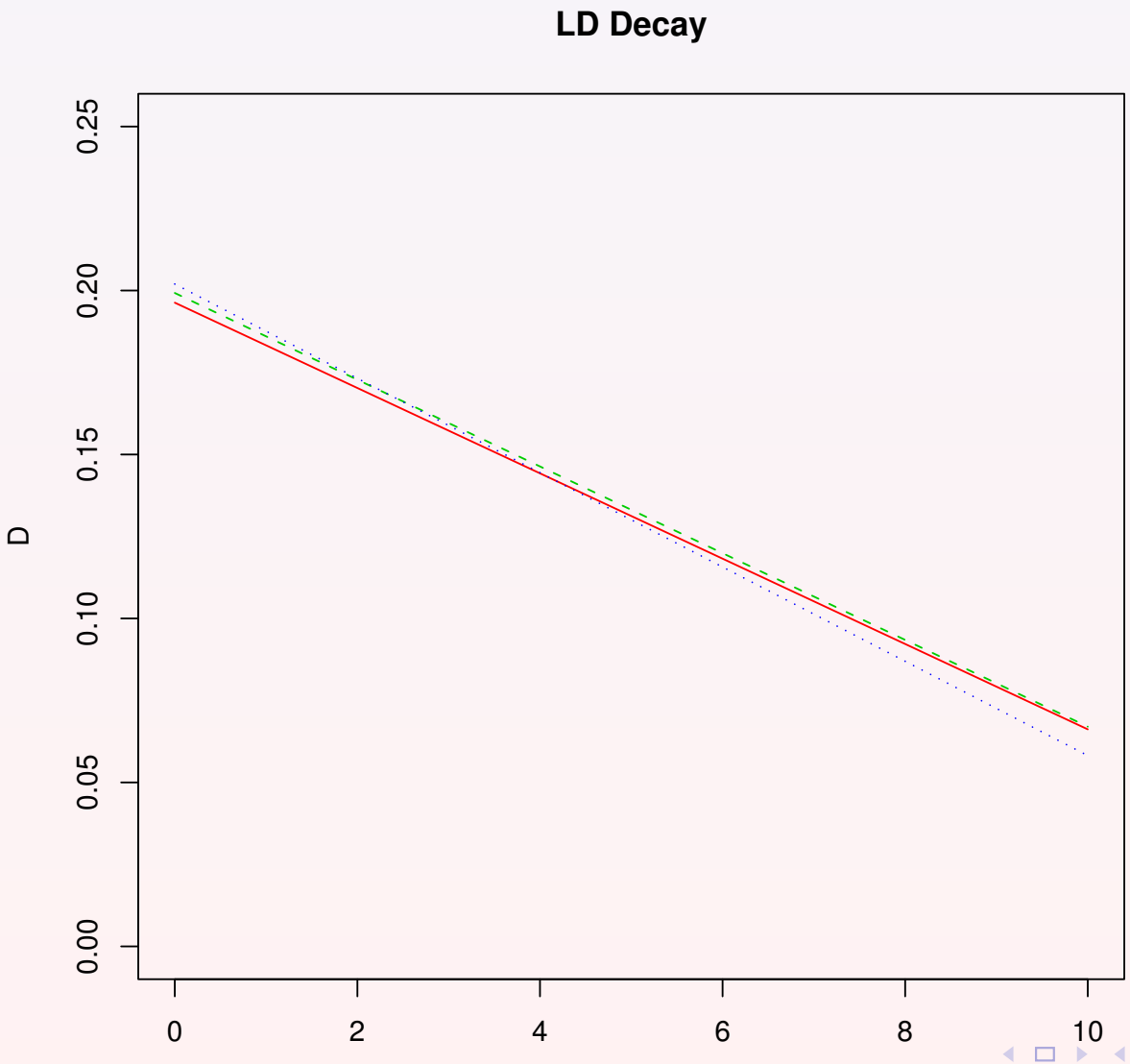
`pyEval` is applied every 10 generations

# Use R to plot

```
>>> from simuPOP import *
>>> from simuRPy import *
>>> simu = simulator(
...     population(size=1000, ploidy=2, loci=[2]),
...     randomMating(),
...     rep = 3)
>>> simu.evolve(
...     preOps = [initByValue([1,2,2,1])],
...     ops = [
...         recombinator(rate=0.1),
...         stat(LD=[0,1]),
...         varPlotter('LD[0][1]', numRep=3, step=10, saveAs='ld
...             ylim=[0,.25], lty=range(1, 4), col=range(2, 5),
...             xlab='generation', ylab='D', title='LD Decay'),
...     ],
...     end=100
... )
True
>>>
```

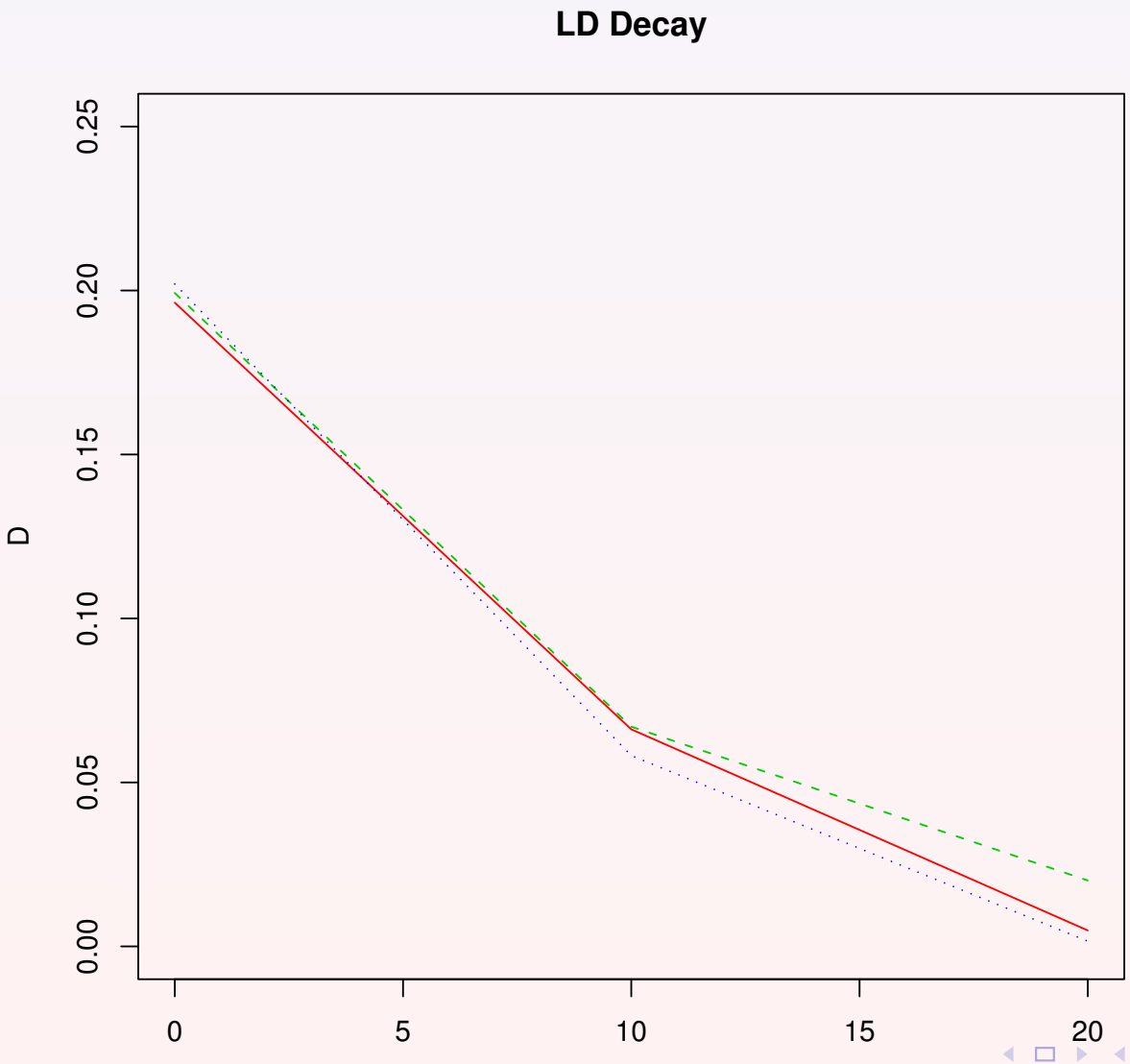
# Evolve!





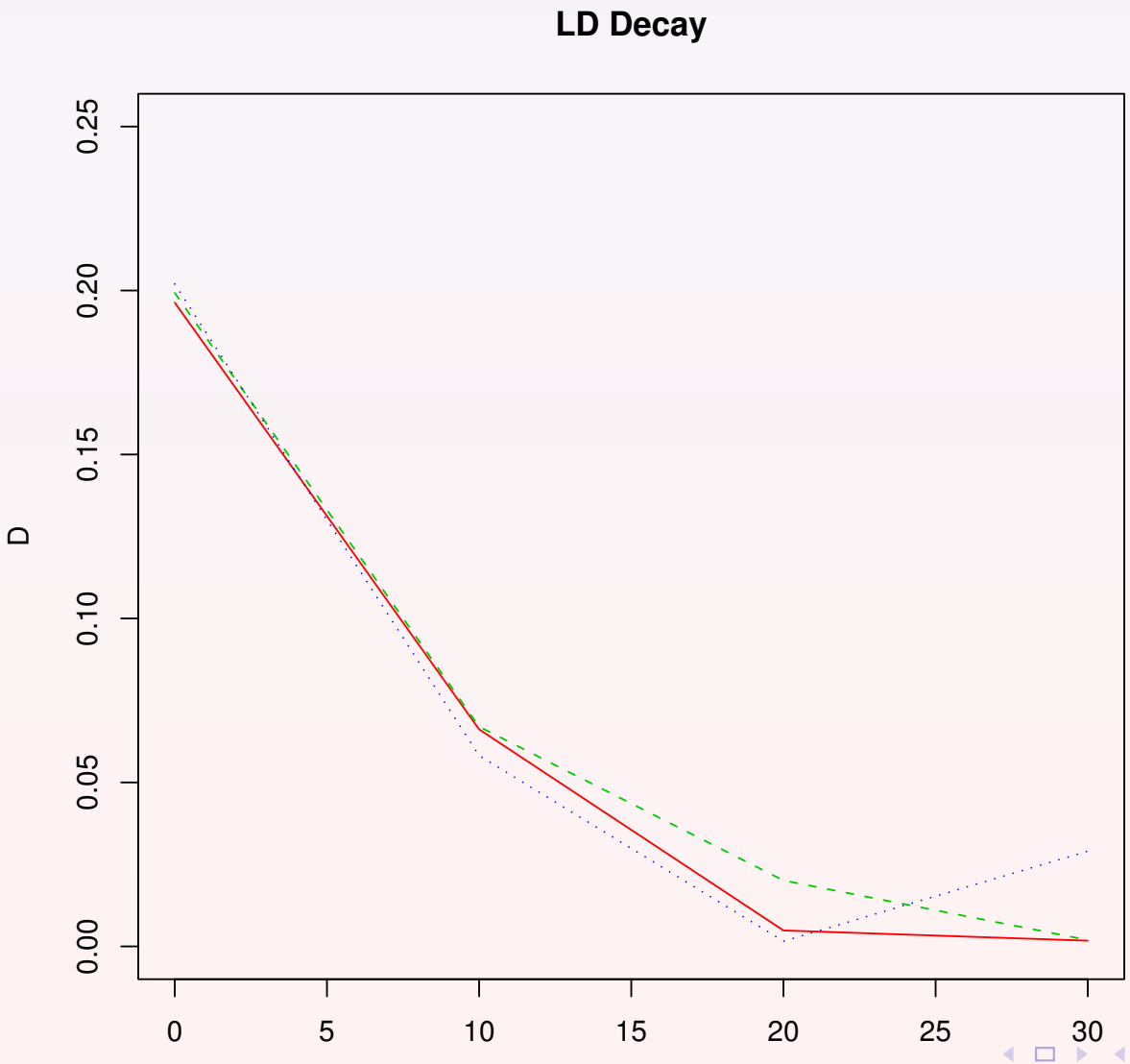

- Update at every 10 generations
- LD=0.25 before generation 0
- LD calculated at the end of each generation

# Evolve!



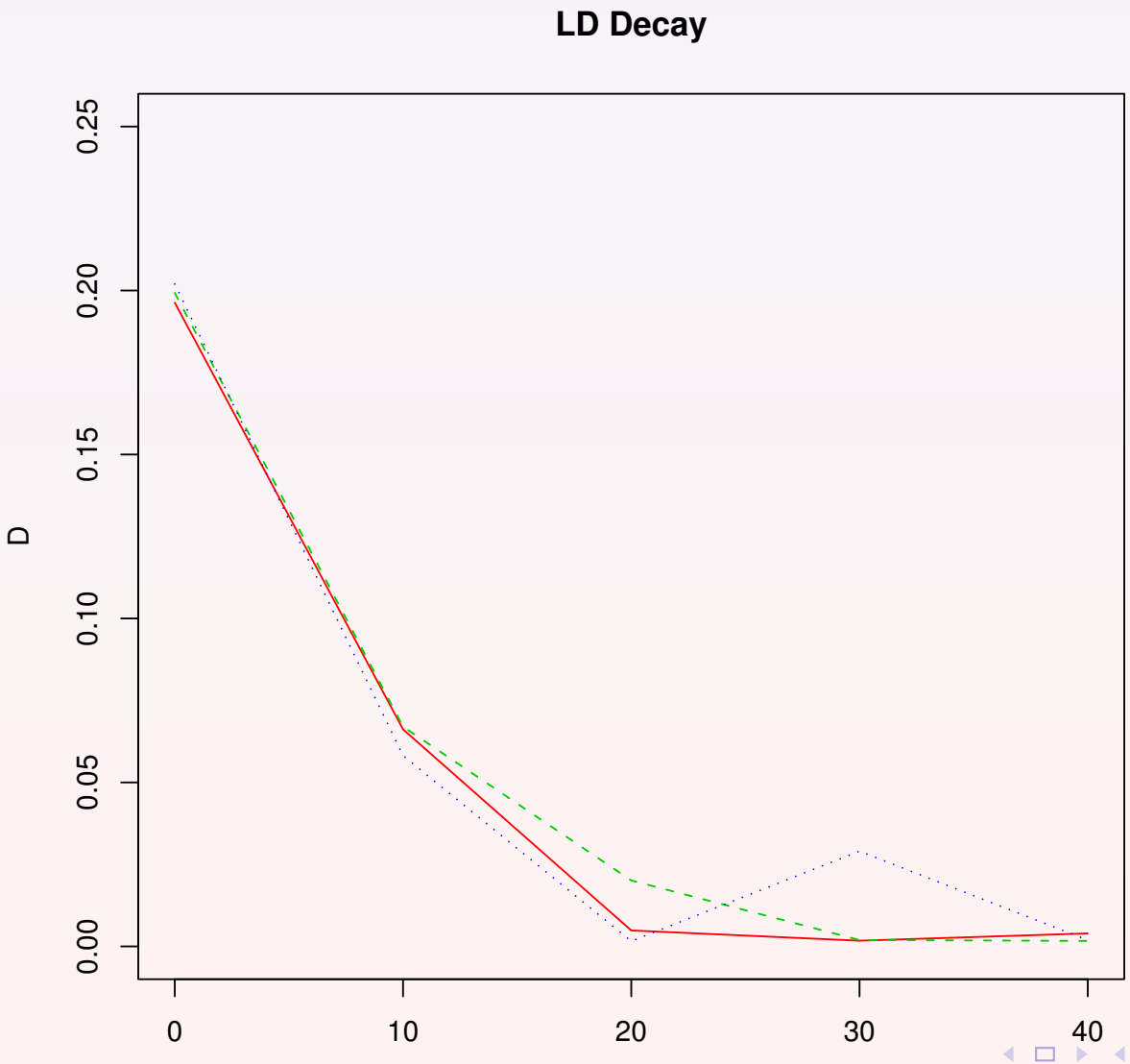

- Update at every 10 generations
- LD=0.25 before generation 0
- LD calculated at the end of each generation

# Evolve!




- Update at every 10 generations
- LD=0.25 before generation 0
- LD calculated at the end of each generation

# Evolve!




- Update at every 10 generations
- LD=0.25 before generation 0
- LD calculated at the end of each generation

# Evolve!

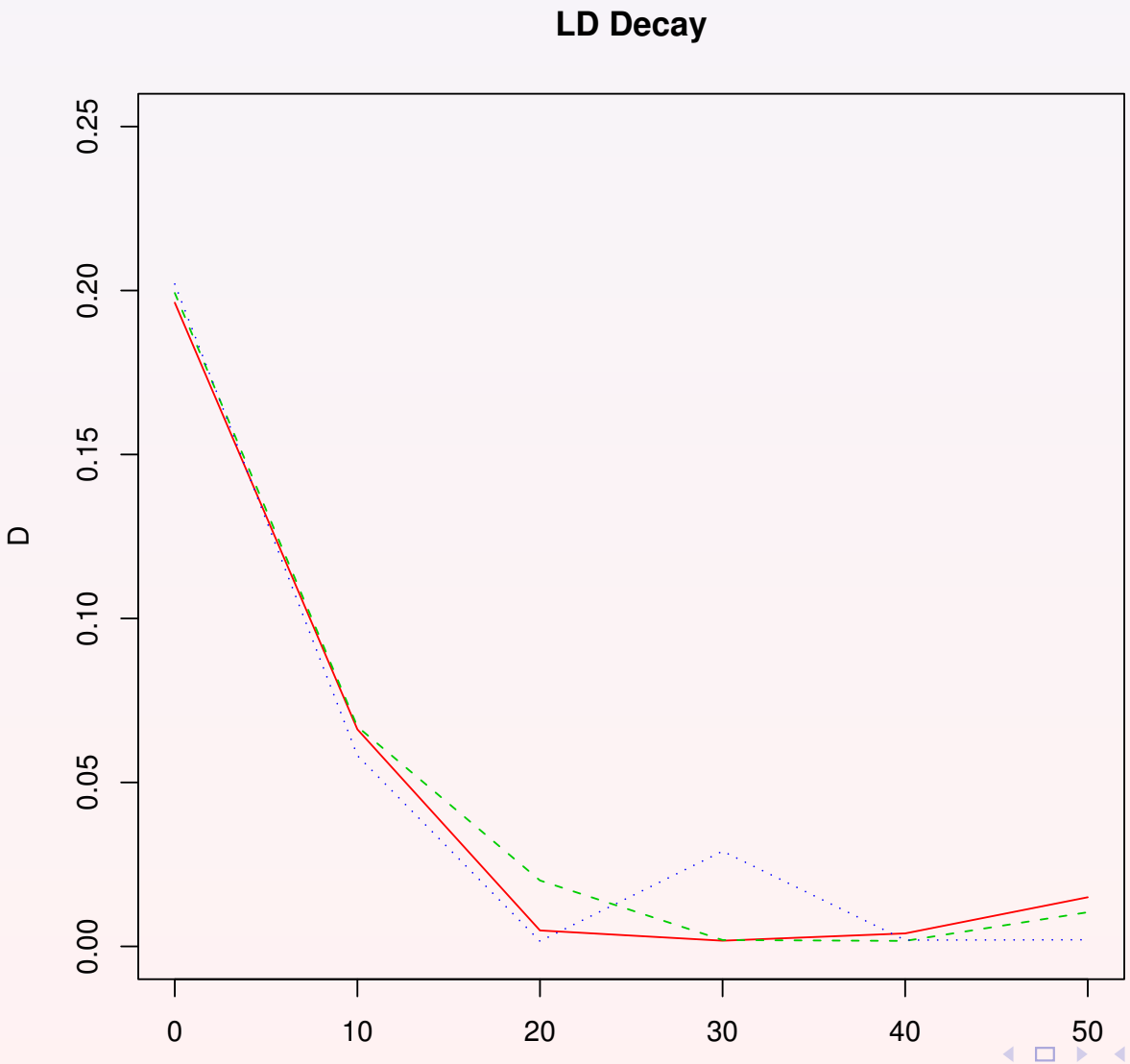
simuPOP tutorial

Bo Peng, Ph.D.

What is simuPOP

An example

simuPOP components




- Update at every 10 generations
- LD=0.25 before generation 0
- LD calculated at the end of each generation

# Evolve!

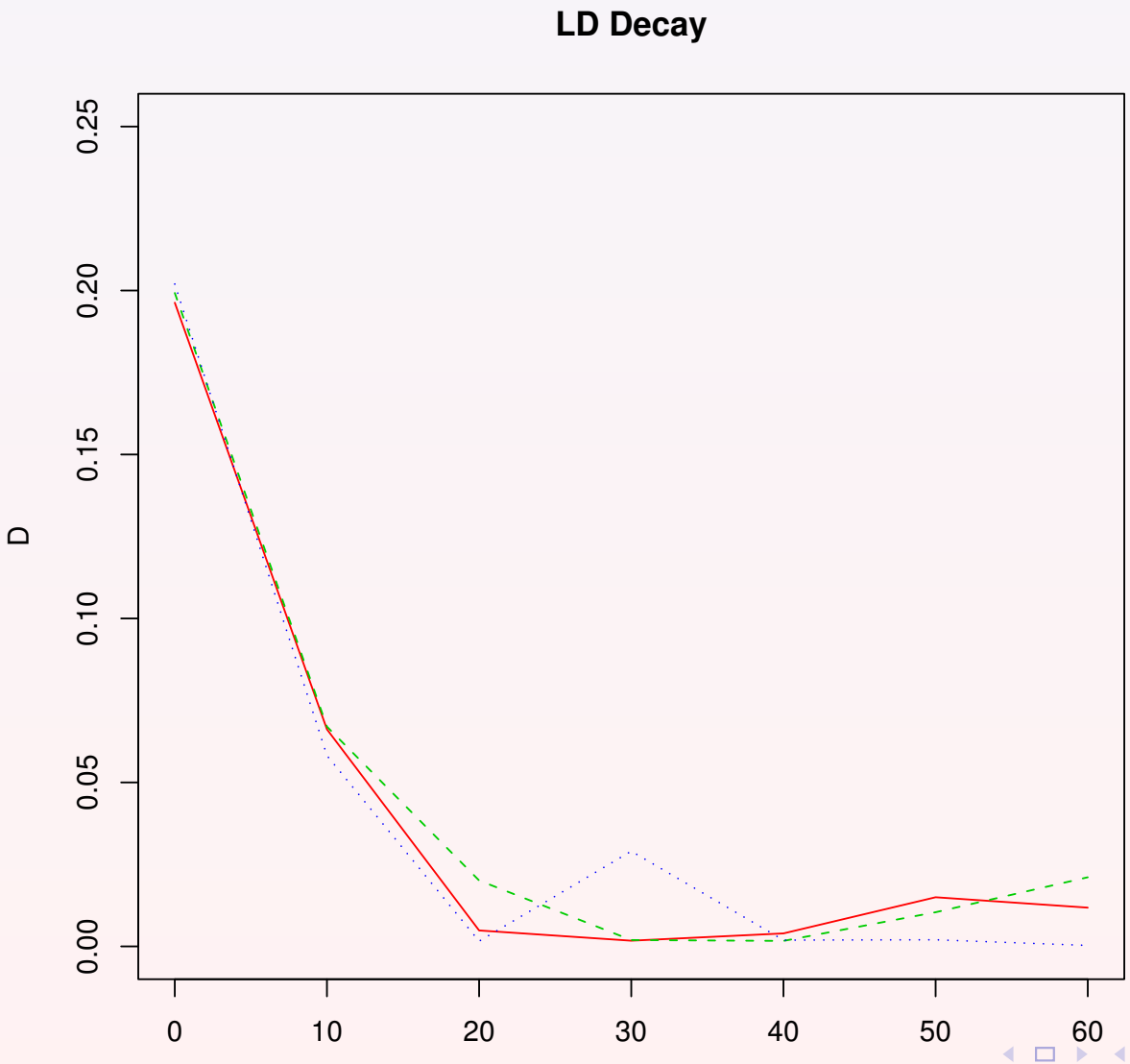
simuPOP tutorial

Bo Peng, Ph.D.

What is simuPOP

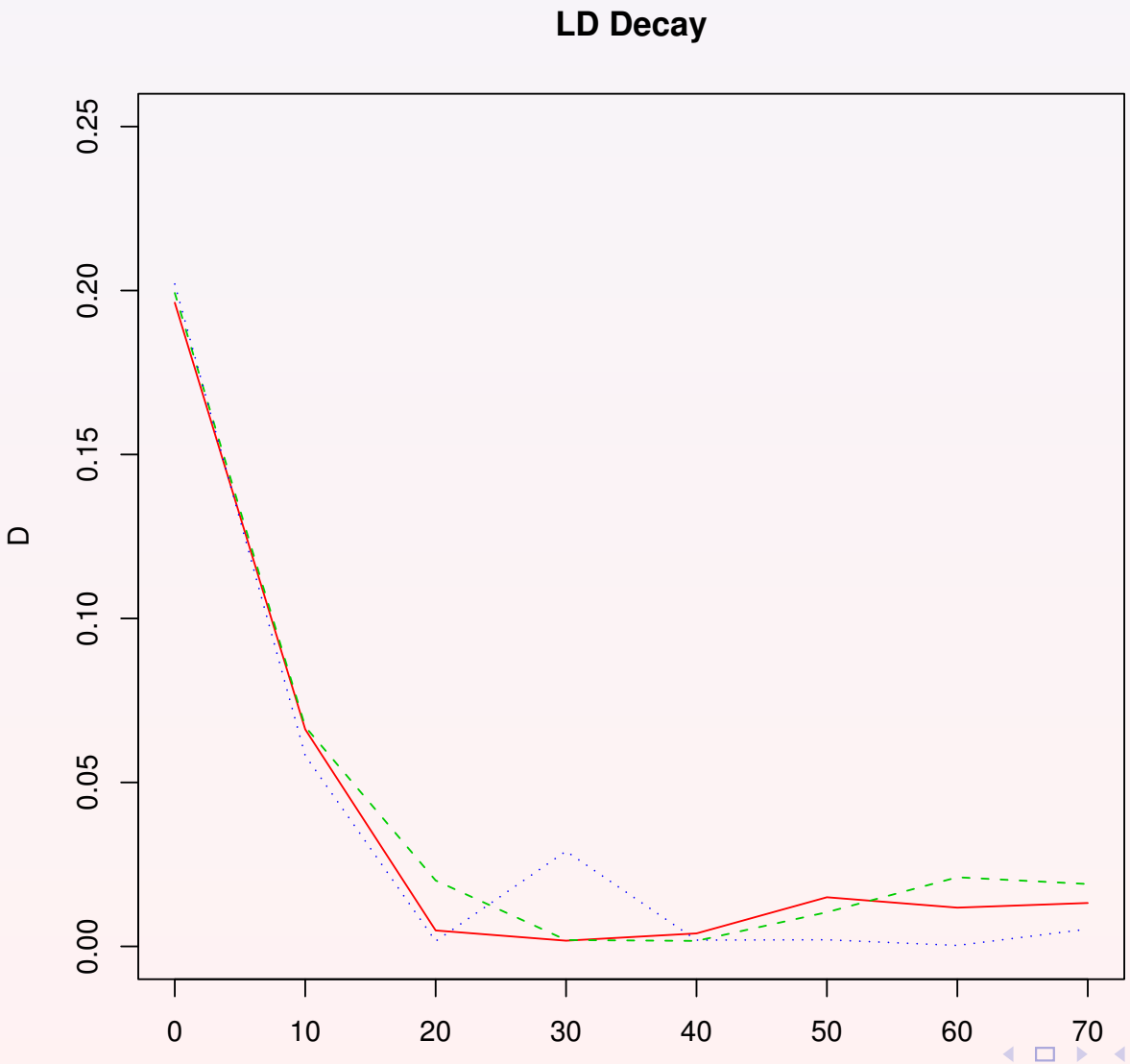
An example

simuPOP components



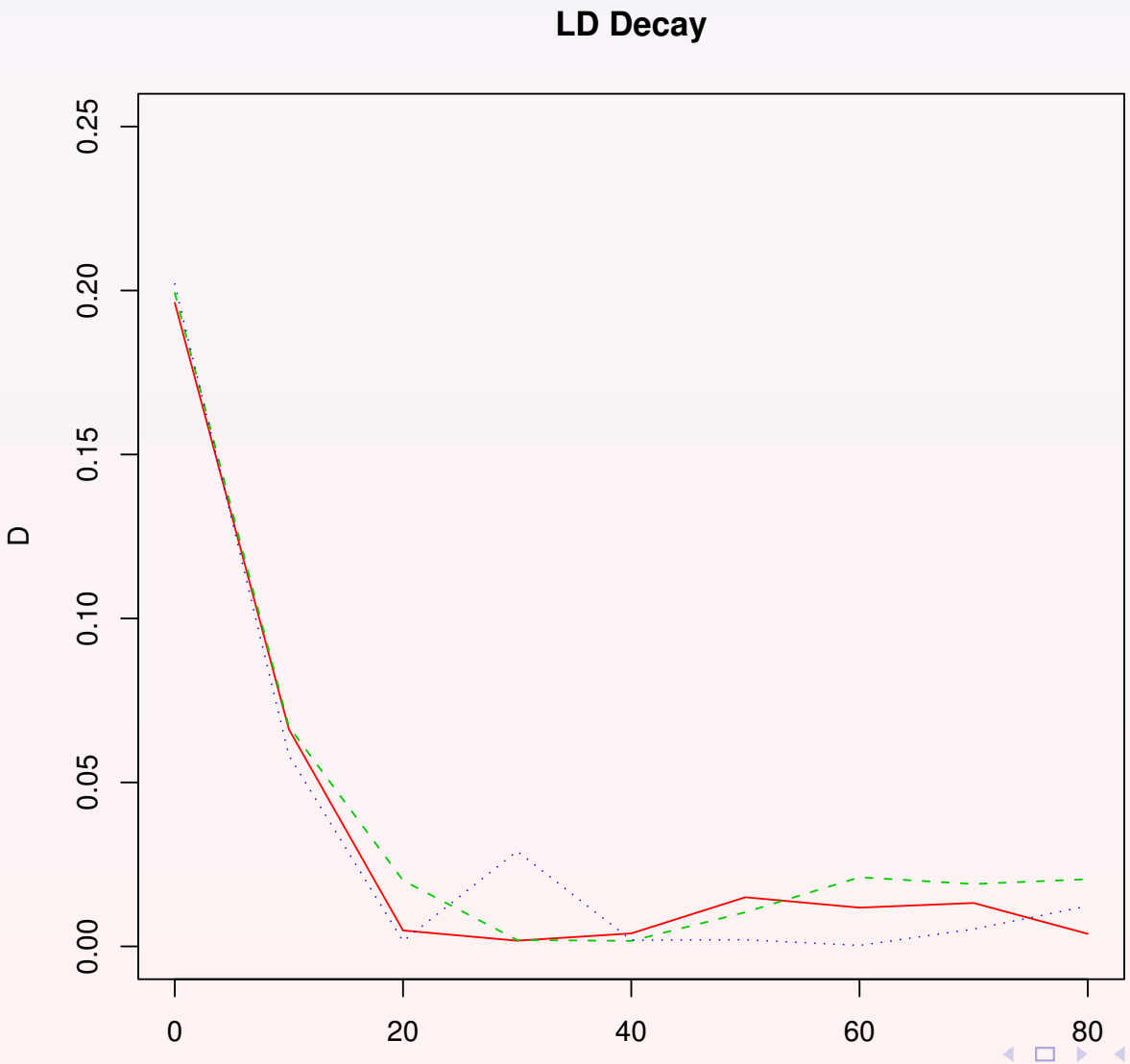

- Update at every 10 generations
- LD=0.25 before generation 0
- LD calculated at the end of each generation

# Evolve!



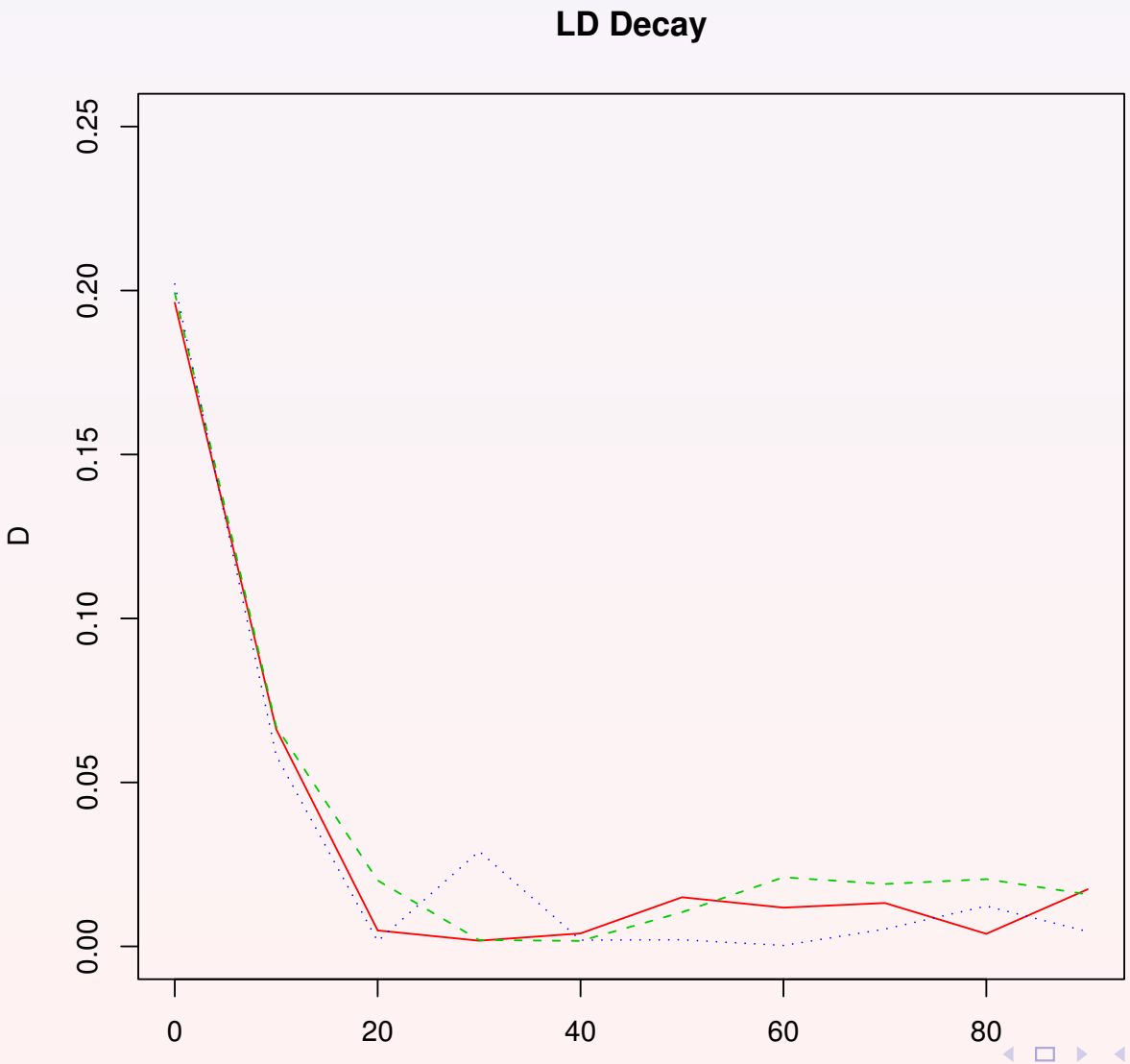

- Update at every 10 generations
- LD=0.25 before generation 0
- LD calculated at the end of each generation

# Evolve!



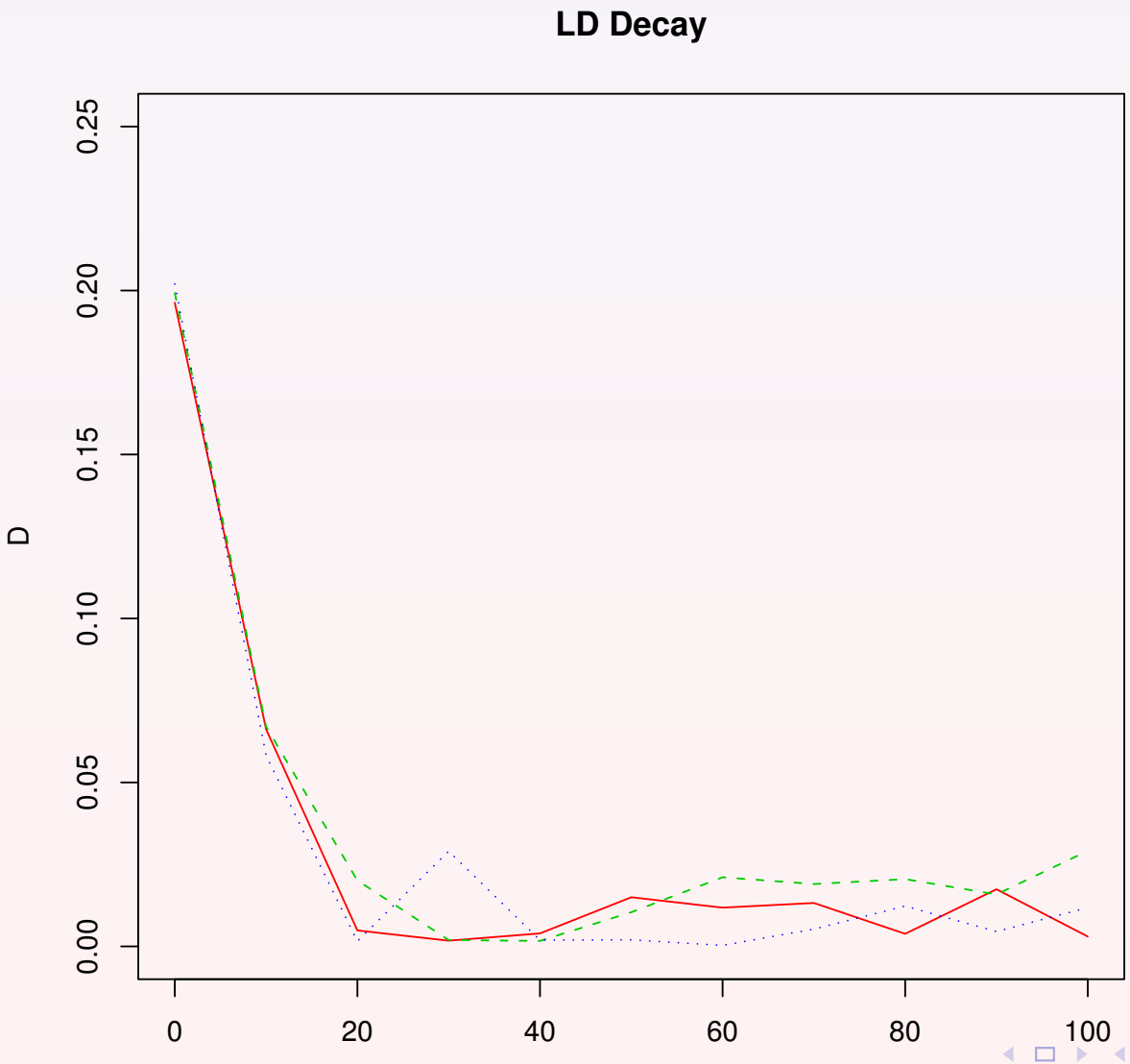

- Update at every 10 generations
- LD=0.25 before generation 0
- LD calculated at the end of each generation

# Evolve!




- Update at every 10 generations
- LD=0.25 before generation 0
- LD calculated at the end of each generation

# Evolve!




- Update at every 10 generations
- LD=0.25 before generation 0
- LD calculated at the end of each generation

# Exercise time

- Start python
- Load simuPOP
- Create a population and run

```
pop.ploidyName()
```

- run `tutorial_example1.py`

# Outline



## 3 simuPOP components

- Population
- Individual
- Operator
- Mating scheme
- Simulator

# Structure of a population

simuPOP tutorial

Bo Peng, Ph.D.

What is simuPOP

An example

simuPOP components

Population

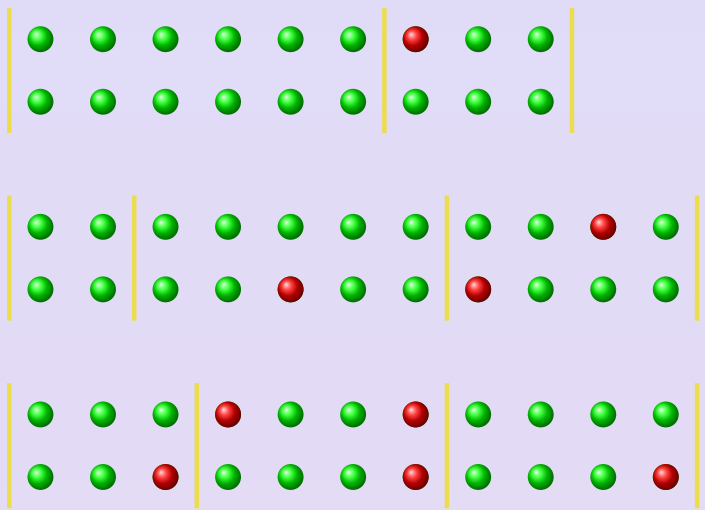
Individual

Operator

Mating scheme

Simulator

- Unaffected
- Affected



gen = 2, numAffected = 5, ...

# Structure of a population

simuPOP tutorial

Bo Peng, Ph.D.

What is simuPOP

An example

simuPOP components

Population

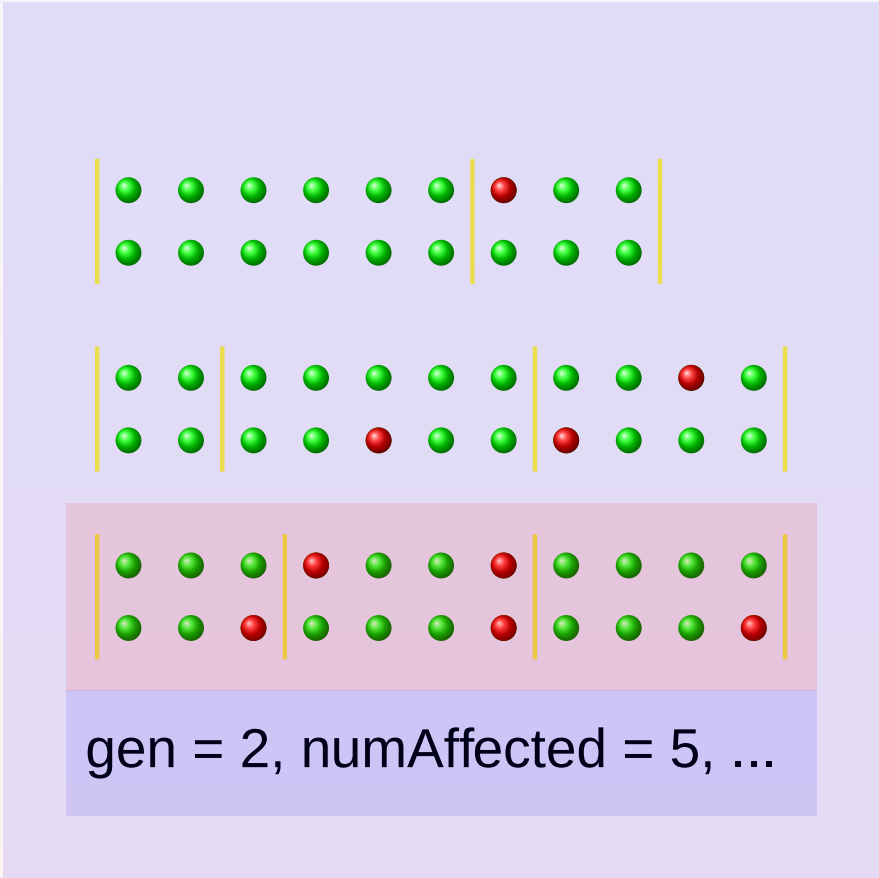
Individual

Operator

Mating scheme

Simulator

- Unaffected
- Affected




Current generation

# Structure of a population

simuPOP tutorial

Bo Peng, Ph.D.

What is simuPOP

An example

simuPOP components

Population

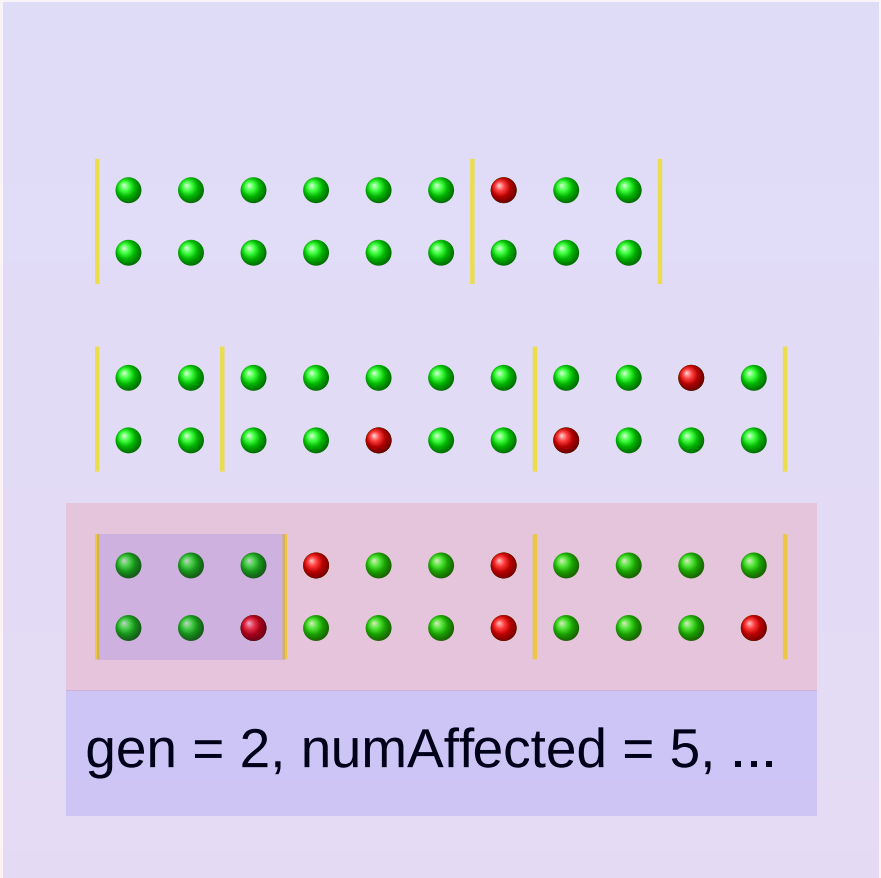
Individual

Operator

Mating scheme

Simulator

- Unaffected
- Affected




Current generation

# Structure of a population

simuPOP tutorial

Bo Peng, Ph.D.

What is simuPOP

An example

simuPOP components

Population

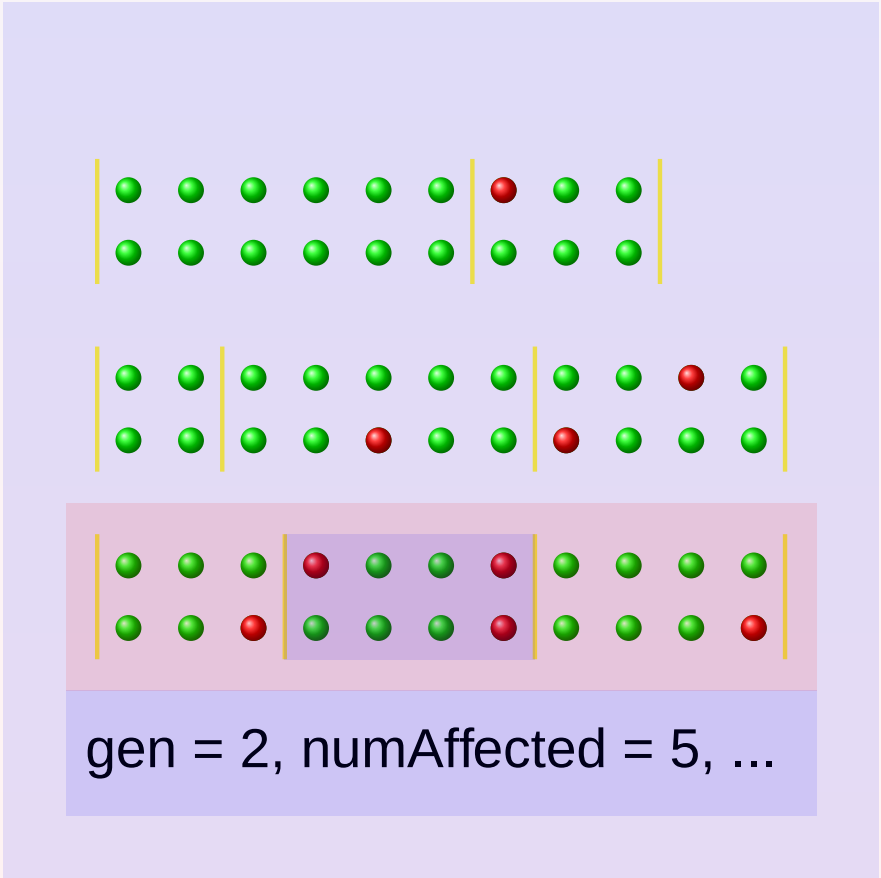
Individual

Operator

Mating scheme

Simulator

- Unaffected
- Affected




Current generation

# Structure of a population

simuPOP tutorial

Bo Peng, Ph.D.

What is simuPOP

An example

simuPOP components

Population

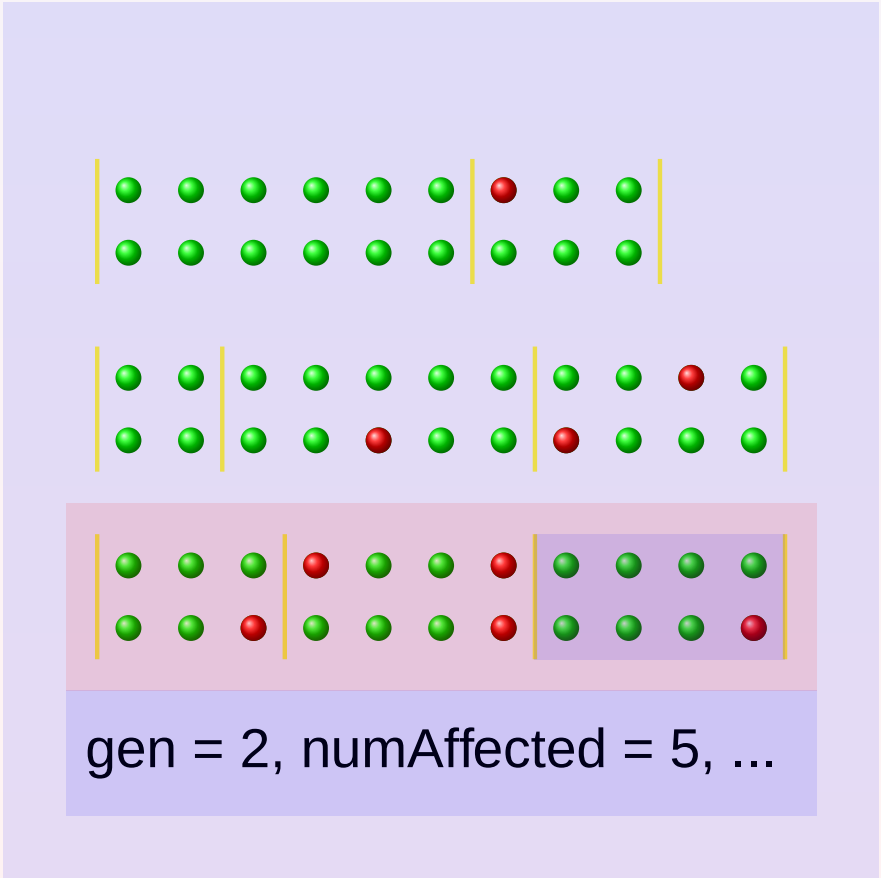
Individual

Operator

Mating scheme

Simulator

- Unaffected
- Affected




Current generation

# Structure of a population

simuPOP tutorial

Bo Peng, Ph.D.

What is simuPOP

An example

simuPOP components

Population

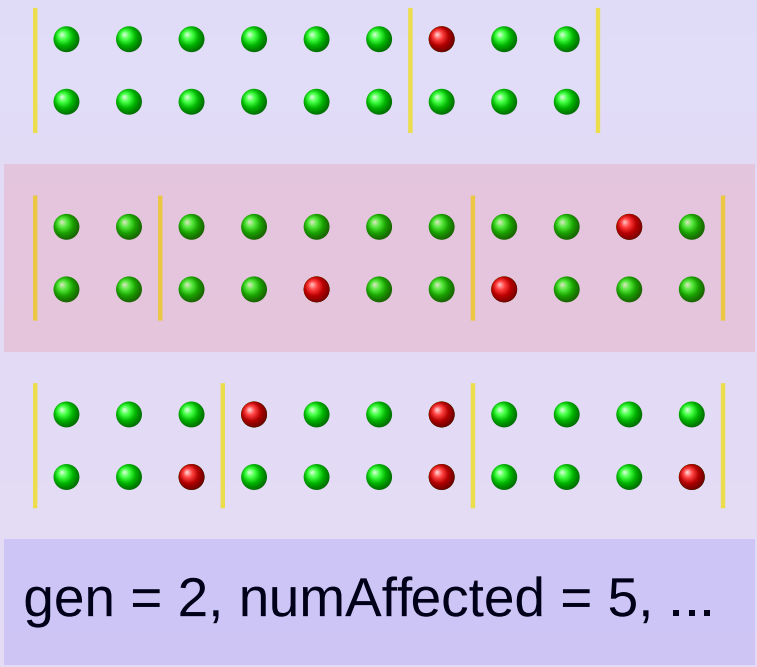
Individual

Operator

Mating scheme

Simulator

- Unaffected
- Affected




Ancestral generation 1

Current generation

# Structure of a population

simuPOP tutorial

Bo Peng, Ph.D.

What is simuPOP

An example

simuPOP components

Population

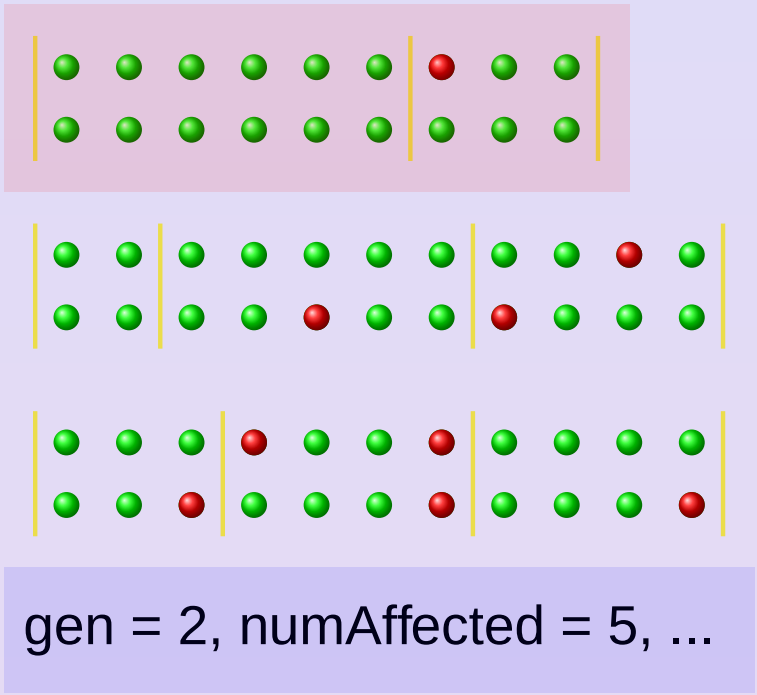
Individual

Operator

Mating scheme

Simulator

- Unaffected
- Affected



# Structure of a population

simuPOP tutorial

Bo Peng, Ph.D.

What is simuPOP

An example

simuPOP components

Population

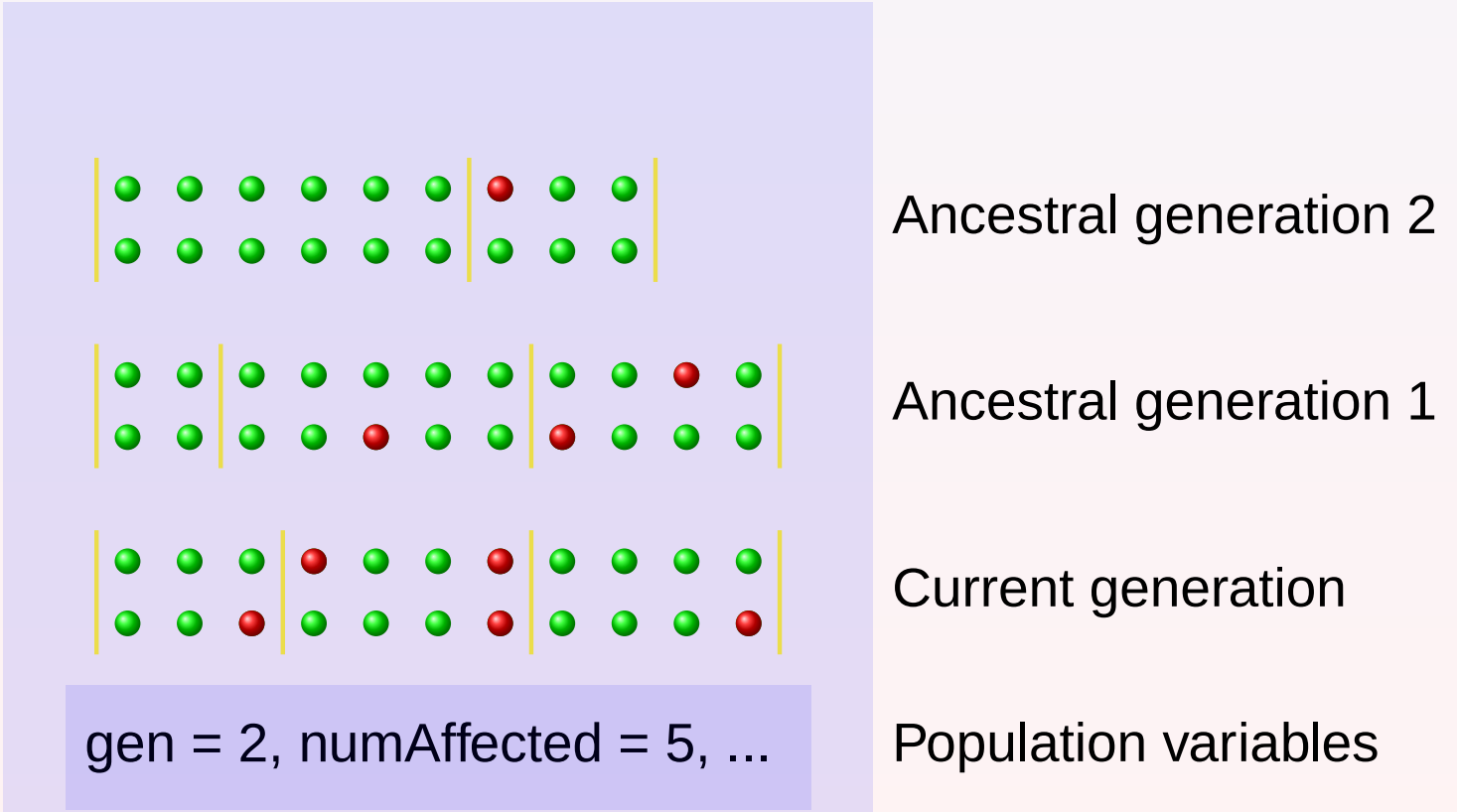
Individual

Operator

Mating scheme

Simulator

- Unaffected
- Affected



# Create and manipulate populations

simuPOP tutorial

Bo Peng, Ph.D.

What is simuPOP

An example

simuPOP components

Population

Individual

Operator

Mating scheme

Simulator

```
>>> pop = population(size=10, loci=[2, 3])
>>> Dump(pop)
Ploidy:                    2
Number of chrom:           2
Number of loci:            2 3
Maximum allele state:      255
Loci positions:
                  1 2
                  1 2 3
Loci names:
                  loc1-1 loc1-2
                  loc2-1 loc2-2 loc2-3
population size:           10
Number of subPop:          1
Subpop sizes:              10
Number of ancestral populations:          0
individual info:
sub population 0:
   0: MU    0  0    0  0  0 |    0  0    0  0  0
   1: MU    0  0    0  0  0 |    0  0    0  0  0
   2: MU    0  0    0  0  0 |    0  0    0  0  0
   3: MU    0  0    0  0  0 |    0  0    0  0  0
```

# Genotypic structure

simuPOP tutorial

Bo Peng, Ph.D.

What is simuPOP

An example

simuPOP components

Population

Individual

Operator

Mating scheme

Simulator

```
>>> pop = population(subPop=[200, 300], loci=[3, 2],
...     maxAllele=3, ploidy=3,
...     lociPos=[[1, 3, 5], [2.5, 4]],
...     alleleNames=['A', 'C', 'T', 'G'])
>>> pop.numLoci(0)
3
>>> pop.totNumLoci()
5
>>> pop.locusPos(4)
4.0
>>> pop.subPopSize(1)
300
>>> pop.popSize()
500
>>> pop.ploidyName()
'triploid'
>>> pop.individual(1).allele(1, 2)
0
>>>
```

# Population manipulation

simuPOP tutorial

Bo Peng, Ph.D.

What is simuPOP

An example

simuPOP components

Population

Individual

Operator

Mating scheme

Simulator

```
>>> # make a copy of pop
>>> pop1 = pop.clone()
>>> # remove loci 2, 3, 4
>>> pop.removeLoci(keep=[0, 1])
>>> # pop2 will have 3 chromosomes, with loci 2, 3, 2
>>> pop2 = MergePopulationsByLoci(pops=[pop, pop1])
>>> # randomly assign alleles using given allele frequencies
>>> InitByFreq(pop2, [0.8, .2])
>>> # calculate population allele frequency
>>> Stat(pop2, alleleFreq=range(pop2.totNumLoci()))
>>> # print allele frequency
>>> print pop2.dvars().alleleFreq
[[0.8066666666666664, 0.1933333333333333], [0.7960000000000000
>>> # assign affection status using a penetrance model
>>> MapPenetrance(pop2, locus=1,
...     penetrance={'0-0': 0.05, '0-1': 0.2, '1-1': 0.8})
>>> # draw case control sample
>>> (sample,) = CaseControlSample(pop2, cases=5, controls=5)
>>> # save sample in Merlin QTDT format
>>> from simuUtil import SaveQTDT
>>> SaveQTDT(sample, output='sample', affectionCode=['U', 'A'],
...     fields=['affection'])
```

# Population manipulation (cont.)

simuPOP tutorial

Bo Peng, Ph.D.

What is simuPOP

An example

simuPOP components

Population

Individual

Operator

Mating scheme

Simulator

```
>>> # have a look at the sample in Merlin-QTDT Format
>>> print open('sample.map').read()
CHROMOSOME MARKER POSITION
1        loc1-1  1.000000
1        loc1-2  3.000000
2        loc1-1_1        1.000000
2        loc1-2_1        3.000000
2        loc1-3  5.000000
3        loc2-1  2.500000
3        loc2-2  4.000000

>>> print open('sample.dat').read()
A        affection
M        loc1-1
M        loc1-2
M        loc1-1_1
M        loc1-2_1
M        loc1-3
M        loc2-1
M        loc2-2
```

# Population manipulation (cont.)

simuPOP tutorial

Bo Peng, Ph.D.

What is simuPOP

An example

simuPOP components

Population

Individual

Operator

Mating scheme

Simulator

```
>>> print open('sample.ped').read()
1 1 0 0 2 A 2 1 2 2 1 1 1 1 2 1 2 1 1 2
2 1 0 0 2 A 2 2 2 2 1 2 1 1 1 1 1 1 1 2
3 1 0 0 1 A 1 2 2 1 2 1 2 1 1 2 2 1 1 1
4 1 0 0 2 A 2 1 2 2 1 1 1 1 1 1 1 1 1 1
5 1 0 0 1 A 1 1 1 2 2 1 1 1 1 1 2 1 1 1
6 1 0 0 1 U 1 1 1 1 2 1 1 1 1 1 1 2 1 2
7 1 0 0 2 U 1 1 1 1 1 1 1 1 2 1 1 1 1 1
8 1 0 0 1 U 1 1 1 2 1 2 2 1 1 1 1 1 2 1
9 1 0 0 2 U 1 1 1 2 2 1 1 2 1 1 2 2 1 1
10 1 0 0 2 U 1 1 1 2 1 1 1 2 1 1 1 1 1 2

>>>
```

# Population variables



```
>>> pop = population(subPop=[5, 10], loci=[5])
>>> InitByFreq(pop, [.6, .3, .1])
>>> Stat(pop, alleleFreq=[1], genoFreq=[2])
>>> print pop.dvars().alleleFreq[1][0]
0.7
>>> from simuUtil import ListVars
>>> ListVars(pop.dvars(), useWxPython=False)
 grp :  -1
 rep :  -1
 alleleNum :
   [1]
     [0]        21
     [1]        7
     [2]        2
 genoFreq :
   [2]
     [0]
       0 :      0.2
       1 :      0.666666666667
       2 :      0.0666666666667
     [1]
       1 :      0.0666666666667
 genoNum :
   [2]
     [0]
       0 :      3.0
       1 :      10.0
       2 :      1.0
     [1]
       1 :      1.0
 alleleFreq :
```

# Population variables (cont.)

simuPOP tutorial

Bo Peng, Ph.D.

What is simuPOP

An example

simuPOP components

Population

Individual

Operator

Mating scheme

Simulator

```
subPop
  [0]
    alleleNum :
      [1]
        [0]    8
        [1]    1
        [2]    1
    genoNum :
      [2]
        [0]
          0 :  2.0
          1 :  2.0
        [1]
          1 :  1.0
    genoFreq :
      [2]
        [0]
          0 :  0.4
          1 :  0.4
        [1]
          1 :  0.2
    alleleFreq :
      [1]
        [0]    0.8
        [1]    0.1
        [2]    0.1
```

# Outline



## 3 simuPOP components

- Population
- Individual
- Operator
- Mating scheme
- Simulator

# Structure of Individuals

Assume ploidy = 2, maxAllele = 1

| 0 | 1 | 2 | | 3 | 4 | | | 5 | 6 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | | 1 | 0 | | | 0 | 1 |
| 0 | 0 | 1 | | 1 | 1 | | | 0 | 1 |

| 0 | | | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
| 0 | | | 1 | 0 | 0 | 0 | 1 |
| 1 | | | 0 | 1 | 1 | 0 | 0 |

Male

- Affected

| fitness | father_id | ··· |
|---|---|---|

# Structure of Individuals

simuPOP tutorial

Bo Peng, Ph.D.

What is simuPOP

An example

simuPOP components

Population

Individual

Operator

Mating scheme

Simulator

Assume ploidy = 2, maxAllele = 1

| 0 | 1 | 2 | | 3 | 4 | | | 5 | 6 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | | 1 | 0 | | | 0 | 1 |
| 0 | 0 | 1 | | 1 | 1 | | | 0 | 1 |

| 0 | | | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
| 0 | | | 1 | 0 | 0 | 0 | 1 |
| 1 | | | 0 | 1 | 1 | 0 | 0 |

Chromosome 0

Male

- Affected

| fitness | father_id | ··· |
|---|---|---|

# Structure of Individuals

Assume ploidy = 2, maxAllele = 1

| 0 | 1 | 2 | | 3 | 4 | | | 5 | 6 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | | 1 | 0 | | | 0 | 1 |
| 0 | 0 | 1 | | 1 | 1 | | | 0 | 1 |

Chromosome 0

| 0 | | | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
| 0 | | | 1 | 0 | 0 | 0 | 1 |
| 1 | | | 0 | 1 | 1 | 0 | 0 |

Chromosome 1

Male

- Affected

| fitness | father_id | ··· |
|---|---|---|

# Structure of Individuals

simuPOP tutorial

Bo Peng, Ph.D.

What is simuPOP

An example

simuPOP components

Population

Individual

Operator

Mating scheme

Simulator

Assume ploidy = 2, maxAllele = 1

| 0 | 1 | 2 | | 3 | 4 | | | 5 | 6 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | | 1 | 0 | | | 0 | 1 |
| 0 | 0 | 1 | | 1 | 1 | | | 0 | 1 |

| 0 | | | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
| 0 | | | 1 | 0 | 0 | 0 | 1 |
| 1 | | | 0 | 1 | 1 | 0 | 0 |

Chromosome 0

Chromosome 1

Male

Sex

- Affected

| fitness | father_id | ··· |
|---|---|---|

# Structure of Individuals

Assume ploidy = 2, maxAllele = 1

| 0 | 1 | 2 | | 3 | 4 | | | 5 | 6 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | | 1 | 0 | | | 0 | 1 |
| 0 | 0 | 1 | | 1 | 1 | | | 0 | 1 |

Chromosome 0

| 0 | | | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
| 0 | | | 1 | 0 | 0 | 0 | 1 |
| 1 | | | 0 | 1 | 1 | 0 | 0 |

Chromosome 1

Male

Sex

- Affected

Affection status

| fitness | father_id | ··· |
|---|---|---|

# Structure of Individuals

Assume ploidy = 2, maxAllele = 1

| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 | 1 |

Chromosome 0

| 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 |

Chromosome 1

Male

Sex

- Affected

Affection status

| fitness | father_id | ... |
|---|---|---|

Information fields

# Individuals

simuPOP tutorial

Bo Peng, Ph.D.

What is simuPOP

An example

simuPOP components

Population

Individual

Operator

Mating scheme

Simulator

```
>>> pop = population(subPop=[5, 8], loci=[5],
...     infoFields=['penetrance'])
>>> InitByFreq(pop, [.6, .3, .1])
>>> MaPenetrance(pop, locus=2, penetrance=[0.05, 0.2, 0.5],
...     wildtype=[0], infoFields=['penetrance'])
>>> # iterate through all inviduals in subPop 1
>>> for ind in pop.individuals(1):
...     print 'Aff: %d Fit: %.3f Geno: %d %d' % \
...         (ind.affected(), ind.info('penetrance'), \
...         ind.allele(2, 0), ind.allele(2, 1))
...
Aff: 0 Fit: 0.500 Geno: 1 1
Aff: 0 Fit: 0.200 Geno: 2 0
Aff: 0 Fit: 0.050 Geno: 0 0
Aff: 0 Fit: 0.050 Geno: 0 0
Aff: 0 Fit: 0.050 Geno: 0 0
Aff: 1 Fit: 0.200 Geno: 2 0
Aff: 0 Fit: 0.200 Geno: 0 1
Aff: 0 Fit: 0.050 Geno: 0 0
>>>
```

# Information fields

simuPOP tutorial

Bo Peng, Ph.D.

What is simuPOP

An example

simuPOP components

Population

Individual

Operator

Mating scheme

Simulator

```
>>> pop = population(100, infoFields=['father_idx', 'mother_idx'
>>> simu = simulator(pop, randomMating(numOffspring=2))
>>> simu.evolve(ops=[parentsTagger()], end=5)
True
>>> ind = simu.population(0).individual(0)
>>> ind1 = simu.population(0).individual(1)
>>> print ind.info('father_idx'), ind.info('mother_idx')
56.0 46.0
>>> print ind1.info('father_idx'), ind1.info('mother_idx')
56.0 46.0
>>>
>>>
```

# Outline



## 3 simuPOP components

- Population
- Individual
- Operator
- Mating scheme
- Simulator

# Stages

# Stages, an example

# Output

# Table-like output

# Outline

simuPOP tutorial

Bo Peng, Ph.D.

What is simuPOP

An example

simuPOP components

Population

Individual

Operator

Mating scheme

Simulator

## 3 simuPOP components

- Population
- Individual
- Operator
- Mating scheme
- Simulator

# Mating schemes

simuPOP tutorial

Bo Peng, Ph.D.

What is simuPOP

An example

simuPOP components

Population

Individual

Operator

Mating scheme

Simulator

# Outline

simuPOP tutorial

Bo Peng, Ph.D.

What is simuPOP

An example

simuPOP components

Population

Individual

Operator

Mating scheme

Simulator

## 3 simuPOP components

- Population
- Individual
- Operator
- Mating scheme
- Simulator

# Simulator