simuPOP tutorial

Bo Peng, Ph.D.

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

simuPOP is ...

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 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...

How to use simuPOP

Just like R/S-Plus or Matlab, you can

- Interactively manipulate populations and evolve them
- Write a script (in Python) and set arbitrarily complicated simulations
- Run existing scripts

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 a population
- study the evolution of (complex) genetic diseases
- simulate samples to validate gene-mapping methods

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??

The core of simuPOP is written in C++, and is provided (wrapped) as Python modules.

- Python is easy to learn
- Python is easy to write and maintain
- Python is

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    0.198508    0.196872    0.203029
 10    0.072649    0.060332    0.047324
 20    0.020258    0.038249    0.003640
 30    0.025511    0.036691    0.004265
 40    0.003195    0.012665    0.009455
 50    0.021836    0.025322    0.002663
 60    0.000849    0.014016    0.005947
 70    0.010480    0.007194    0.016178
 80    0.002101    0.001977    0.021530
 90    0.017101    0.008740    0.018721
100    0.006286    0.013090    0.028828
```

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!

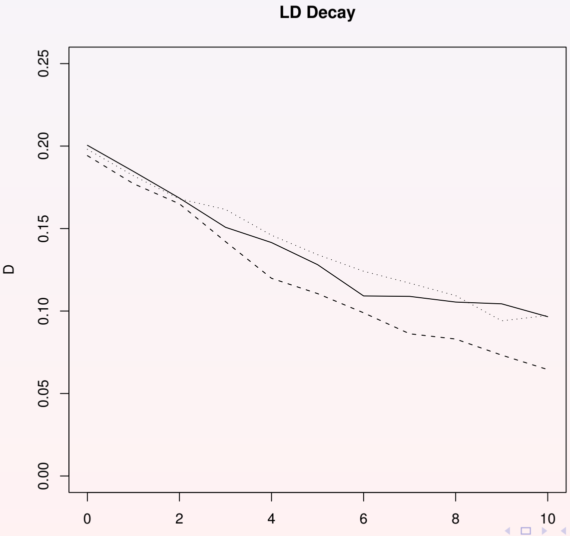

```
>>> from simuPOP import *
>>> simu = simulator(
...     population(size=1000, ploidy=2, loci=[2]),
...     randomMating(),
...     rep = 3)
>>> simu.evolve(
...     preOps = [initByValue([1,2,2,1])],
...     ops = [
...         recombinator(rate=0.1),
...         stat(LD=[0,1]),
...         pyEval(r"'%3d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

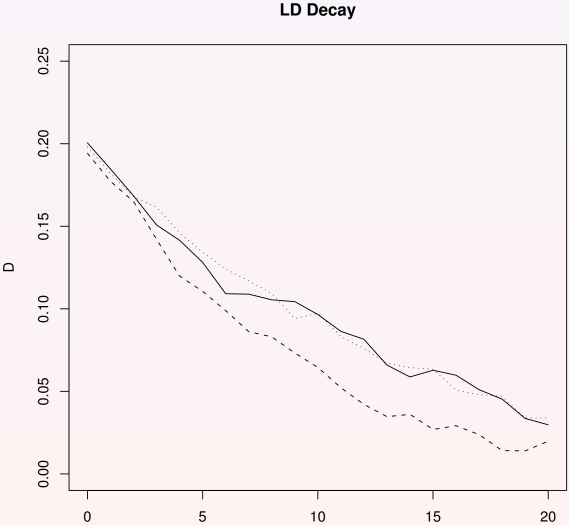

```
>>> from simuPOP import *
>>> from simuRPy import *
>>> simu = simulator(
...     population(size=1000, ploidy=2, loci=[2]),
...     randomMating(),
...     rep = 3)
>>> simu.evolve(
...     preOps = [initByValue([1,2,2,1])],
...     ops = [
...         recombinator(rate=0.1),
...         stat(LD=[0,1]),
...         varPlotter('LD[0][1]', numRep=3, step=10,
...                    ylim=[0,.25], xlab='generation',
...                    ylab='D', title='LD Decay'),
...     ],
...     end=100
... )
True
>>> r.dev_print(file='log/LDdecay.eps')
{'X11': 2}
>>>
```

Evolve!



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

Evolve!



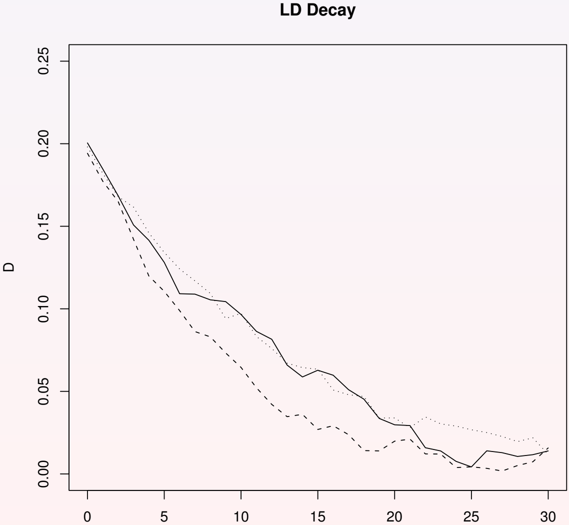
- Update at every 10 generations
- LD=0.25 before generation 0
- LD calculated at the end of each generation

Evolve!

simuPOP tutorial

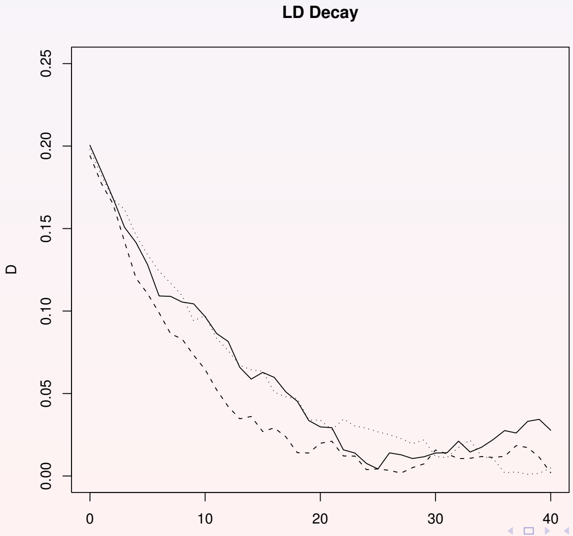
Bo Peng, Ph.D.

- What is simuPOP
- **An example**
- simuPOP components
- Population manipulation



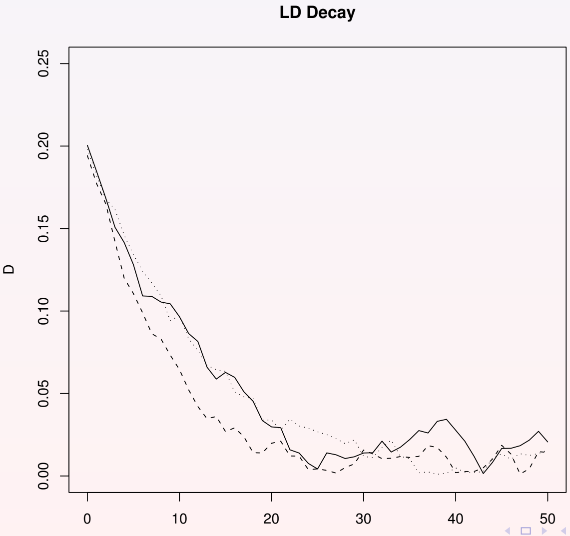
- Update at every 10 generations
- LD=0.25 before generation 0
- LD calculated at the end of each generation

Evolve!



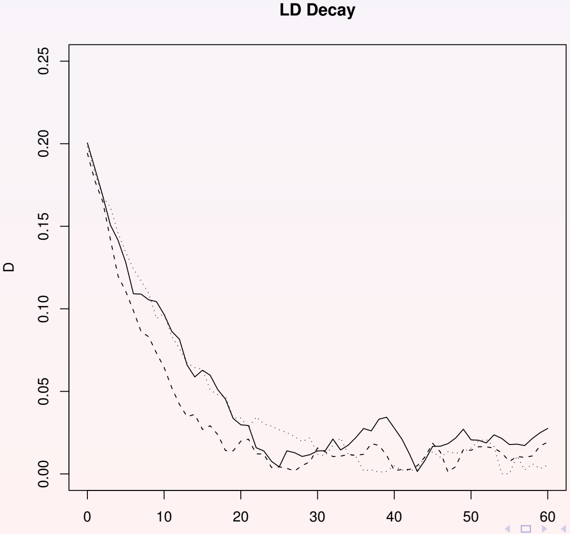
- Update at every 10 generations
- LD=0.25 before generation 0
- LD calculated at the end of each generation

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

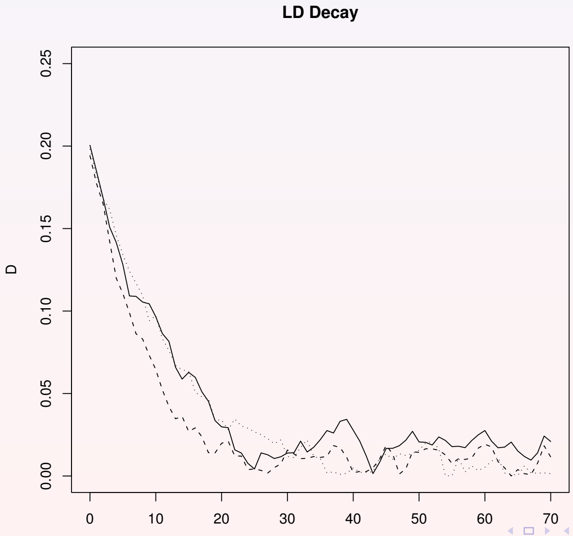
MD Anderson Cancer Center

Evolve!

simuPOP tutorial

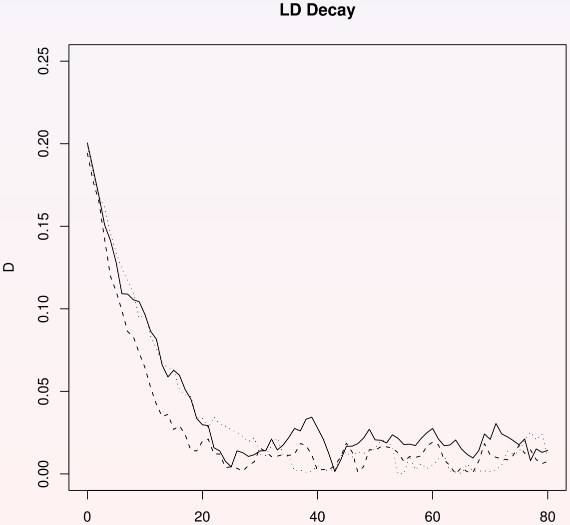
Bo Peng, Ph.D.

- What is simuPOP
- An example
- simuPOP components
- Population manipulation



- 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

simuPOP tutorial

Bo Peng, Ph.D.

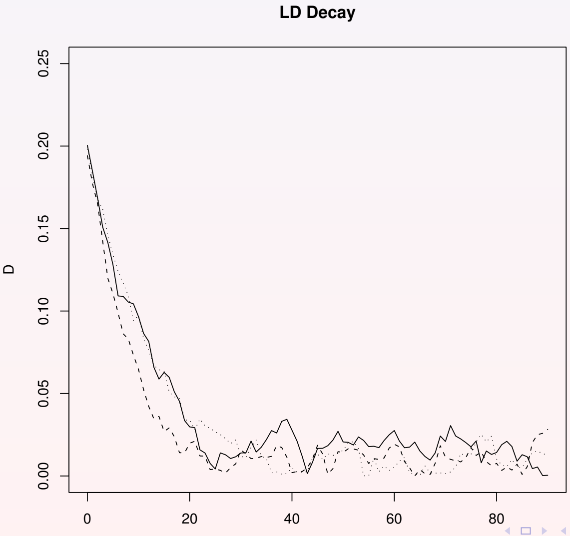
What is simuPOP

An example

simuPOP components

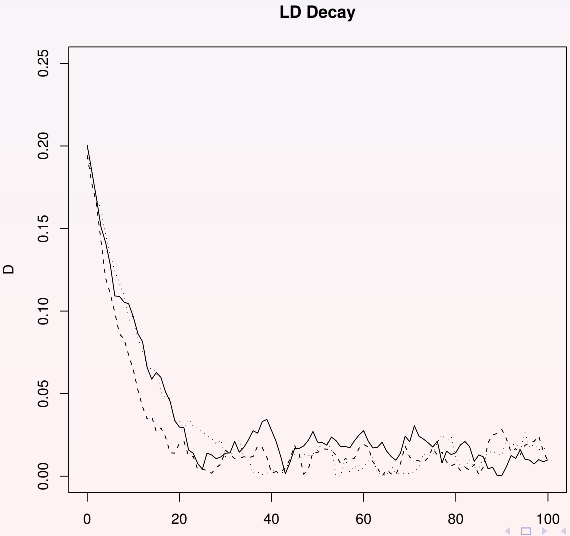
Population manipulation

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`

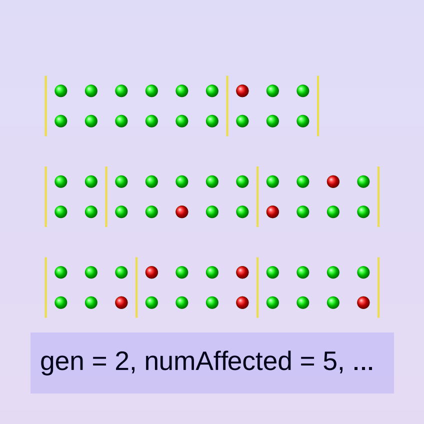
Outline

3 **simuPOP components**

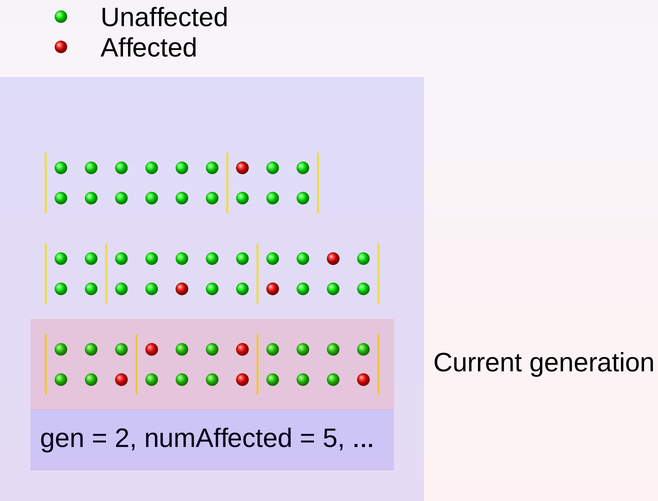
- Population object
- Operators
- Mating scheme, Simulator and forward-time simulation

Structure of a population

- Unaffected
- Affected

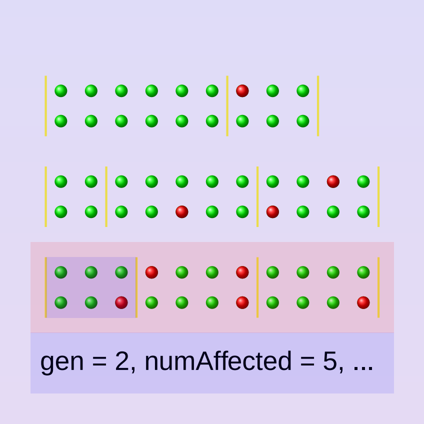


Structure of a population



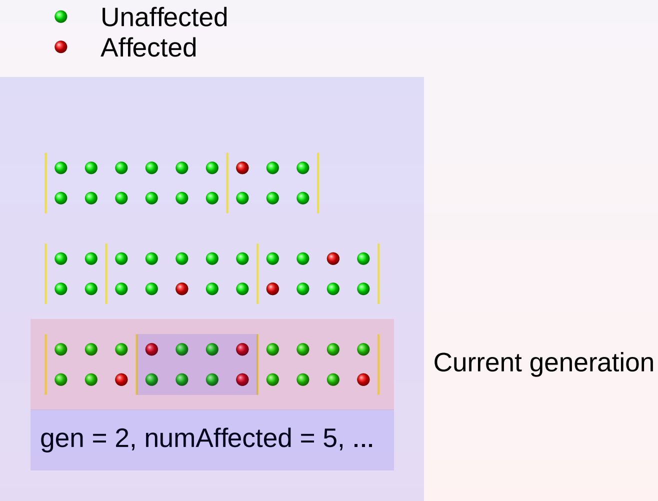
Structure of a population

- Unaffected
- Affected



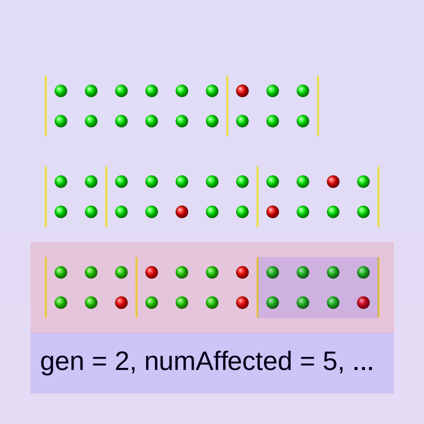
Current generation

Structure of a population



Structure of a population

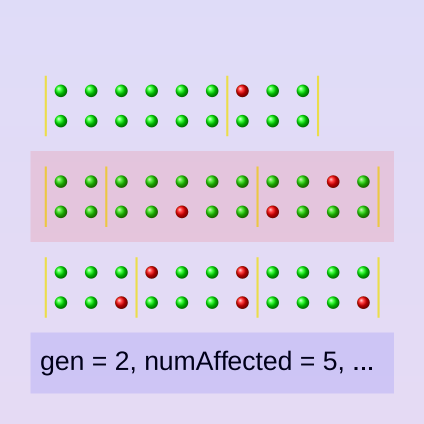
- Unaffected
- Affected



Current generation

Structure of a population

- Unaffected
- Affected

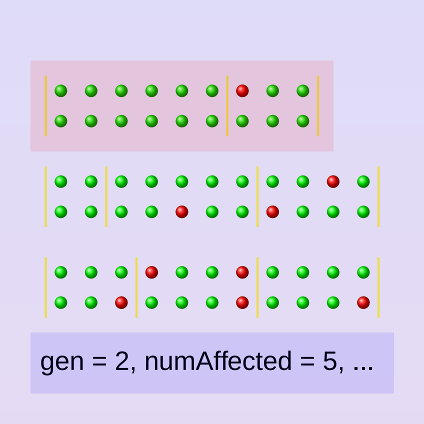


Ancestral generation 1

Current generation

Structure of a population

- Unaffected
- Affected



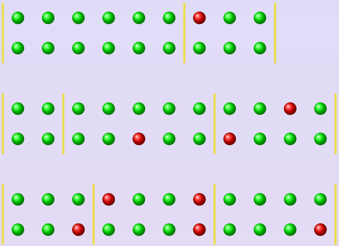
Ancestral generation 2

Ancestral generation 1

Current generation

Structure of a population

- Unaffected
- Affected



Ancestral generation 2

Ancestral generation 1

Current generation

gen = 2, numAffected = 5, ...

Population variables

Create and manipulate populations

Genotypic structure

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

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

Male

● Affected

fitness	father_id	···

Structure of Individuals

Assume ploidy = 2, maxAllele = 1

Chromosome 0

0	1	2	3	4	5	6
0	1	1	1	0	0	1
0	0	1	1	1	0	1

Chromosome 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 object
- Operators
- Mating scheme, Simulator and forward-time simulation

Population manipulation

Assume ploidy = 2, maxAllele = 1

Chromosome 0

0	1	2	3	4	5	6
0	1	1	1	0	0	1
0	0	1	1	1	0	1

Chromosome 1

0	1	2	3	4	5
0	1	0	0	0	1
1	0	1	1	0	0

Sex: Male

Affected

fitness	father_id	···

Structure of Individuals

Assume ploidy = 2, maxAllele = 1

Chromosome 0

0	1	2	3	4	5	6
0	1	1	1	0	0	1
0	0	1	1	1	0	1

Chromosome 1

0	1	2	3	4	5
0	1	0	0	0	1
1	0	1	1	0	0

Sex: Male

Affection status: 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	···

Information fields

Population strcuture

Information fields

Variables

Outline

simuPOP tutorial

Bo Peng, Ph.D.

What is simuPOP

An example

simuPOP components

Population object

Operators

Mating scheme, Simulator and forward-time simulation

Population manipulation

Stages

Stages, an example

Output

simuPOP tutorial

Bo Peng, Ph.D.

What is simuPOP

An example

simuPOP components

Population object

Operators

Mating scheme, Simulator and forward-time simulation

Population manipulation

Table-like output

Outline

3 **simuPOP components**

- Population object
- Operators
- Mating scheme, Simulator and forward-time simulation

Mating schemes

Simulator

Evolve?!

Outline

4 **Population manipulation**

- Handling of HapMap data