High-dimensional data-sets and the problems they cause

Paul Marjoram, Dept. of Preventive Medicine, Keck School of Medicine, Univ. of Southern California, Los Angeles.

What we do for a living

- Given data D,
- Parameter(s) θ,
- Model M.

- Wish to make inference re. $f(\theta|D)$.
- $f(\theta|D) = f(D|\theta)\,\underbrace{\pi(\theta)}_{\text{Prior}} \,/\, \underbrace{P(D)}_{\text{Normalizing constant}}$

The problem

- Data D,
- Parameter(s) θ,
- Model M

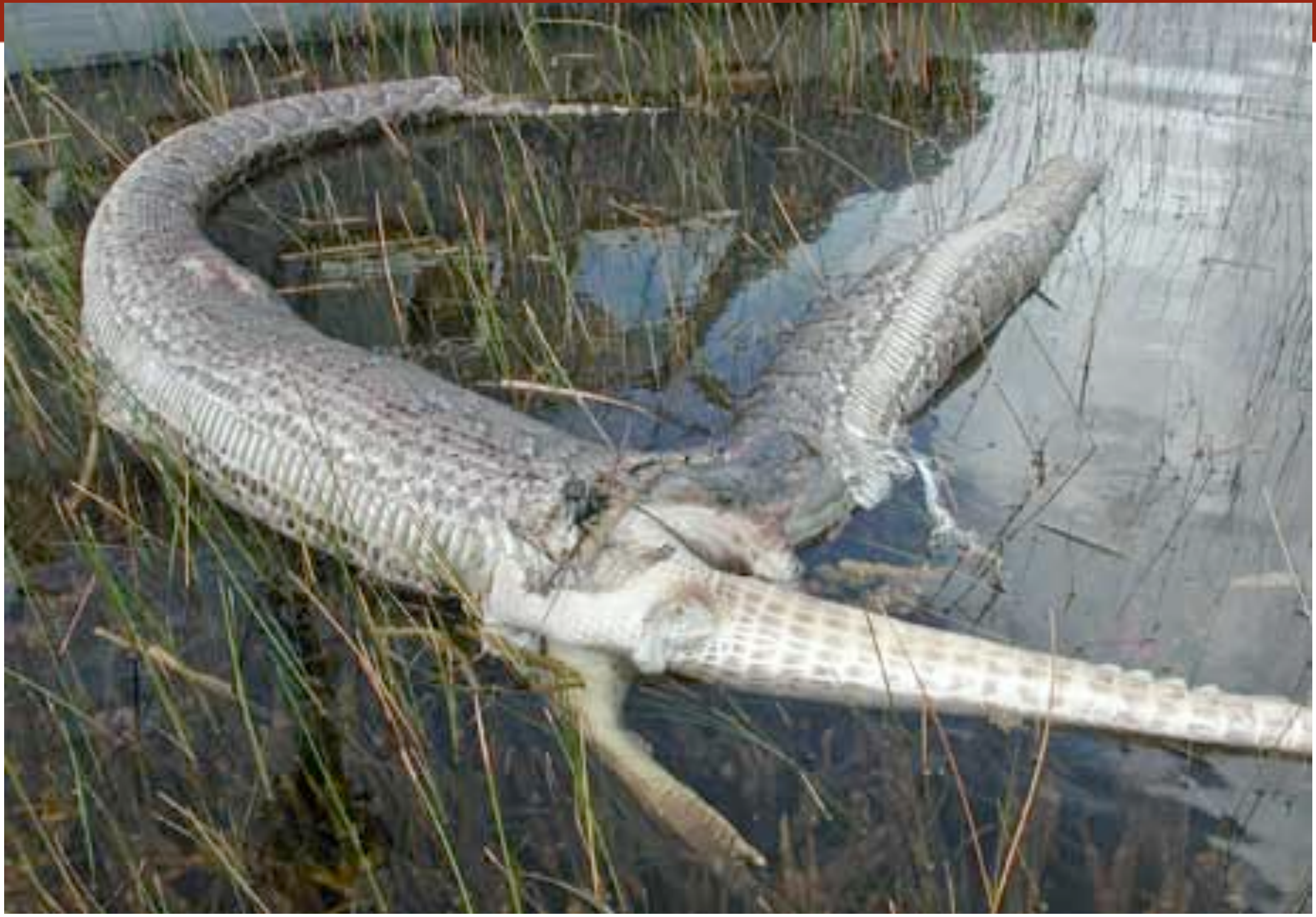
The problem

- Data D,
- Parameter(s) θ,
- Model M

The problem

- Data D,
- Parameter(s) θ,
- Model M

www.GenerallyAwesome.com



National Geographic: September 5, 2006—Unfortunately for a 13-foot (4-meter) Burmese python in Florida's Everglades National Park, eating the enemy seems to have caused the voracious reptile to bust a gut—literally.
Wildlife researchers with the South Florida Natural Resources Center found the dead, headless python in October 2005 after it apparently tried to digest a 6-foot-long (2-meter-long) American alligator. The mostly intact dead gator was found sticking out of a hole in the midsection of the python, and wads of gator skin were found in the snake's gastrointestinal tract.

Summary

- Data sets are growing much larger.
- Larger implies more complex.
- Traditional analysis methods may fail or become computationally intractable. [$f(D|\theta)$]

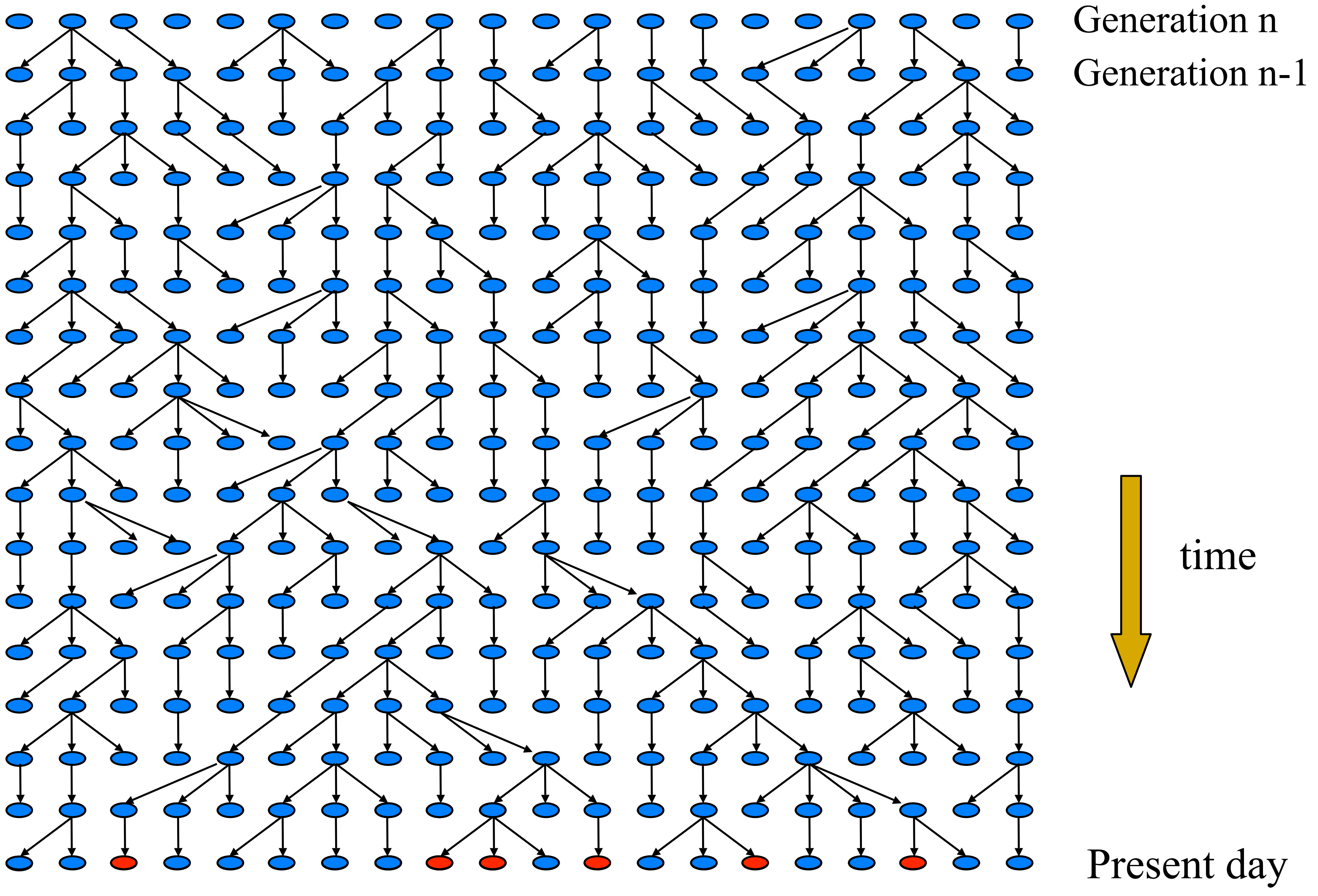
- Possible response:
  - Construct better theory
  - Use simpler (less realistic) models;
  - 'Approximate' methods.

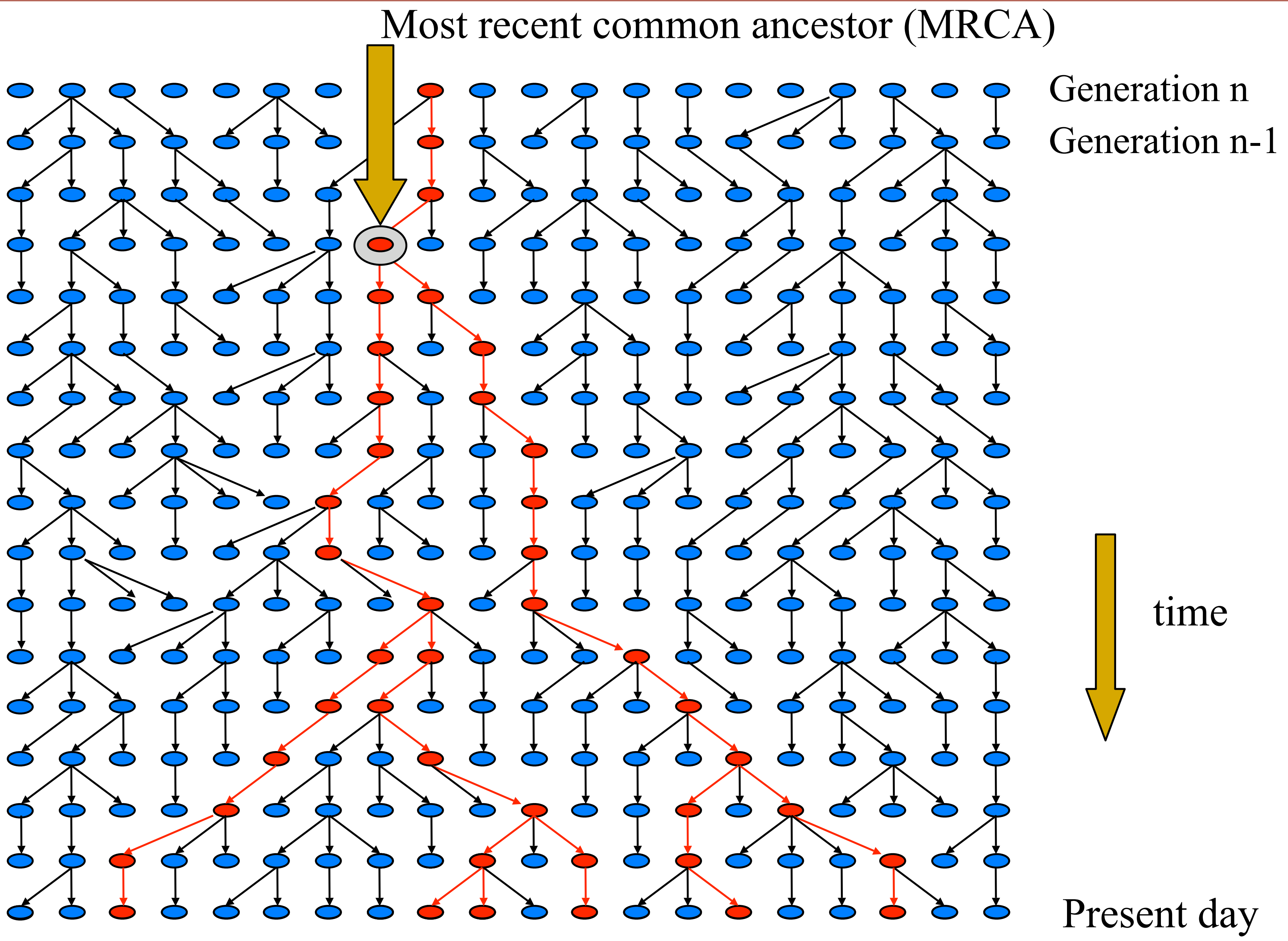
- **Part I - Approximating the model**

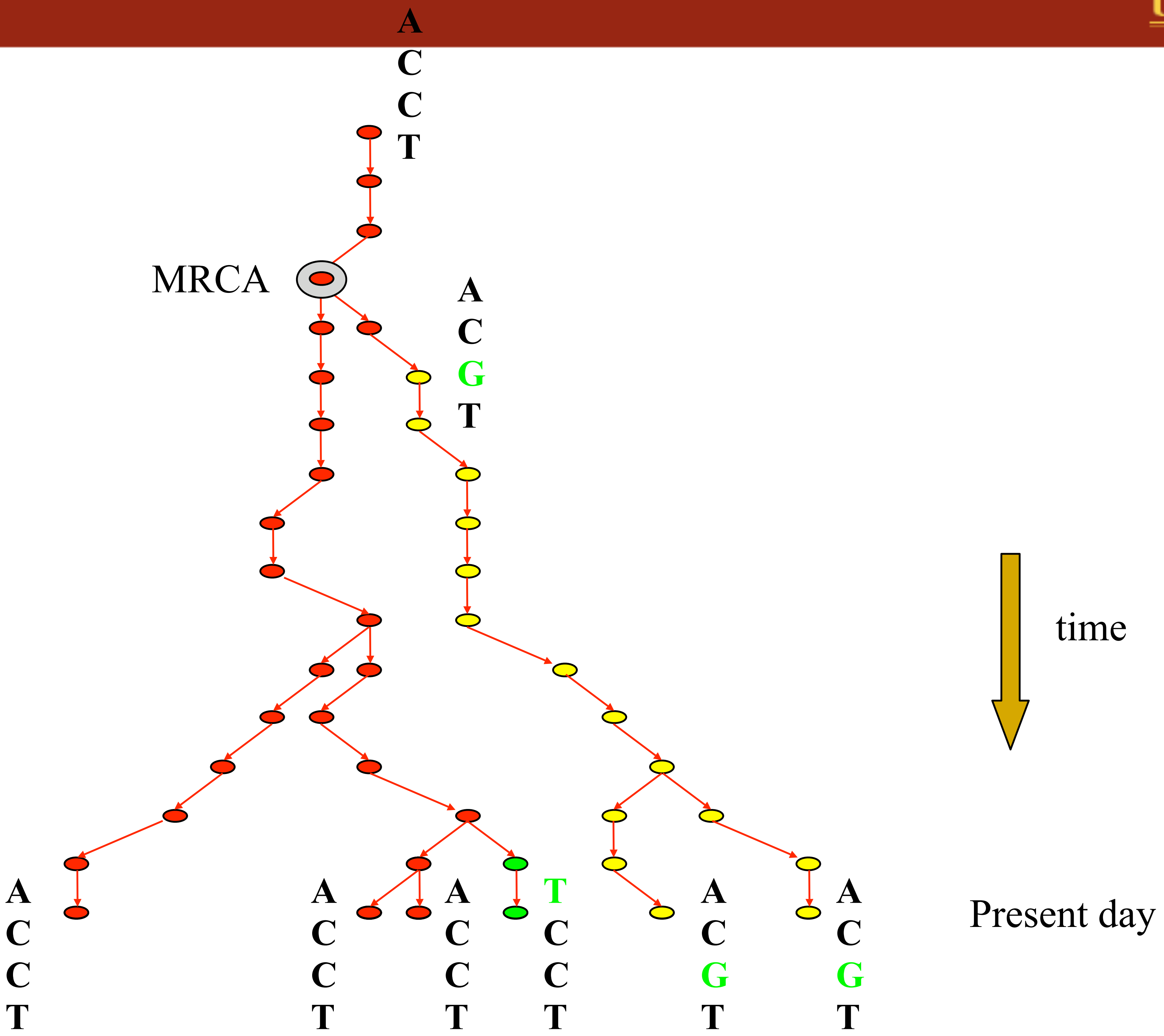
- **Part II - Approximating the model**

All models are wrong; some are useful (Box)

- Recurring example: the coalescent


Generation n
Generation n-1
time
Present day


Most recent common ancestor (MRCA)
Generation n
Generation n-1
time
Present day


A
C
C
T
MRCA
A
C
G
T
time
A
C
C
T
A
C
C
T
A
C
C
T
T
C
C
T
A
C
G
T
A
C
G
T
Present day

Ancestral methods with no recombination (haploid data)

A stochastic (Markov) process.
Time between events is exponentially distributed
As we look back in time **two events** may occur:

i. Two lines of ancestry will **coalesce** to form a single line of ancestry, with prob. $(k-1)/(k-1+\theta)$ where there are currently k lines and $\theta/2$ represents the mutation rate. (Pick a random pair of lines)

ii. A **mutation** will occur to a line of ancestry, changing the type of a gene, with prob. $\theta/(k-1+\theta)$. (Pick a random line)

The process continues until there is a single line of ancestry: the most recent common ancestor (MRCA) of the sample.

A graphical representation of a recombination event that occurs between the 4th and 5th markers.

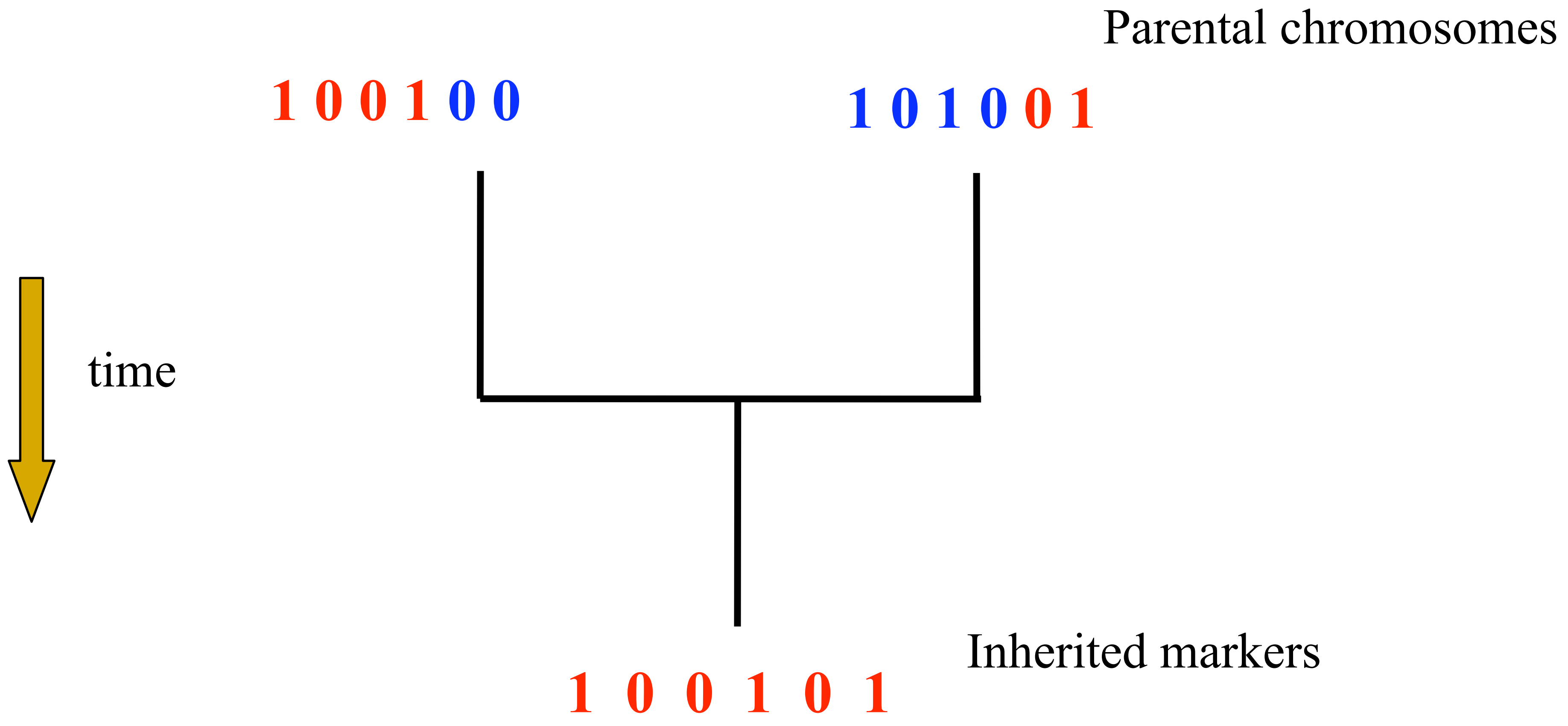
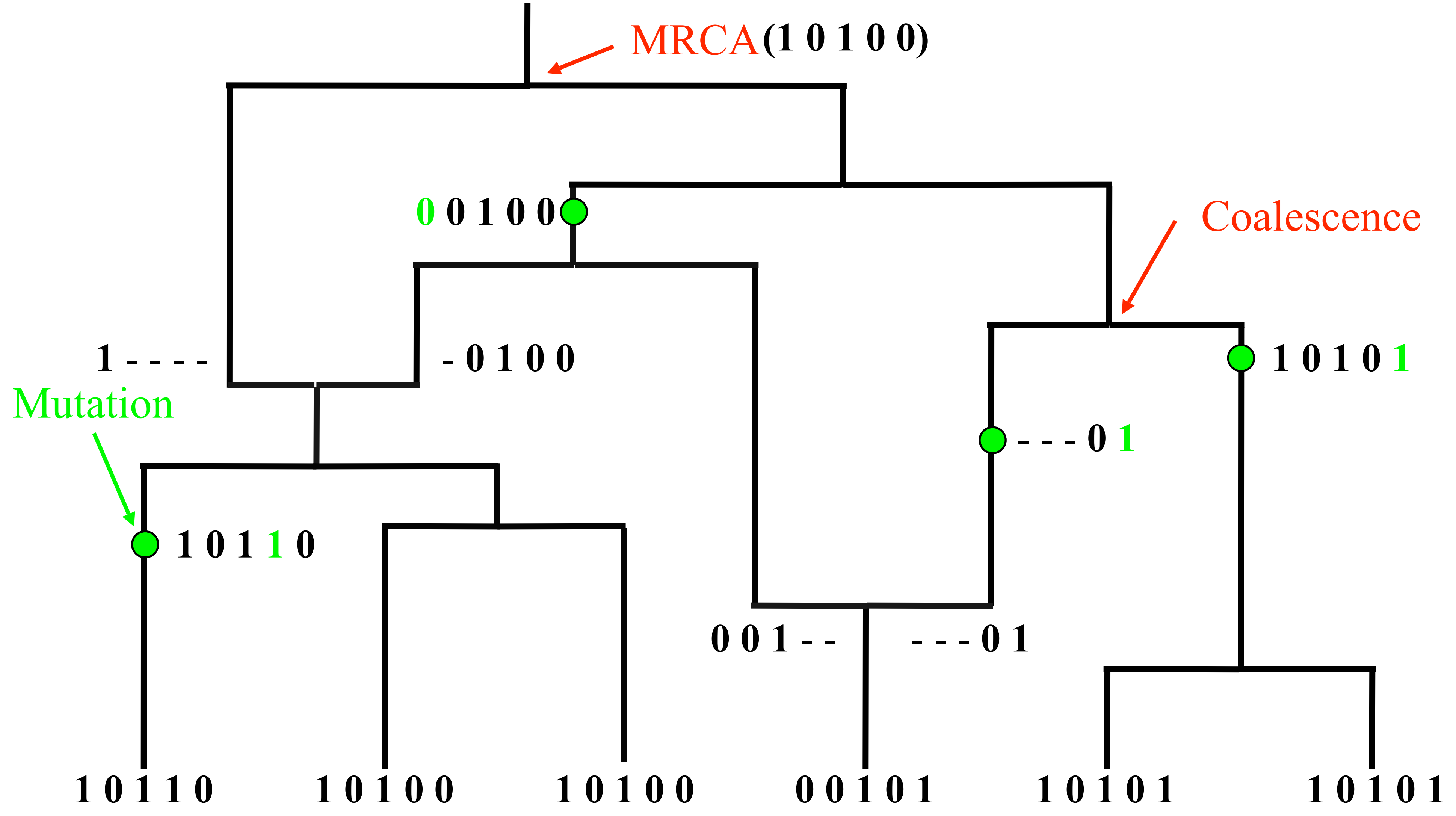


Figure 5: Representation of an ancestry for markers subject to recombination



We trace the ancestry of a sample of 6 marker sequences, until we reach the MRCA. Mutational events are marked in green. (Markers not ancestral to the sample are marked '-')

Coalescent with recombination (diploid data)

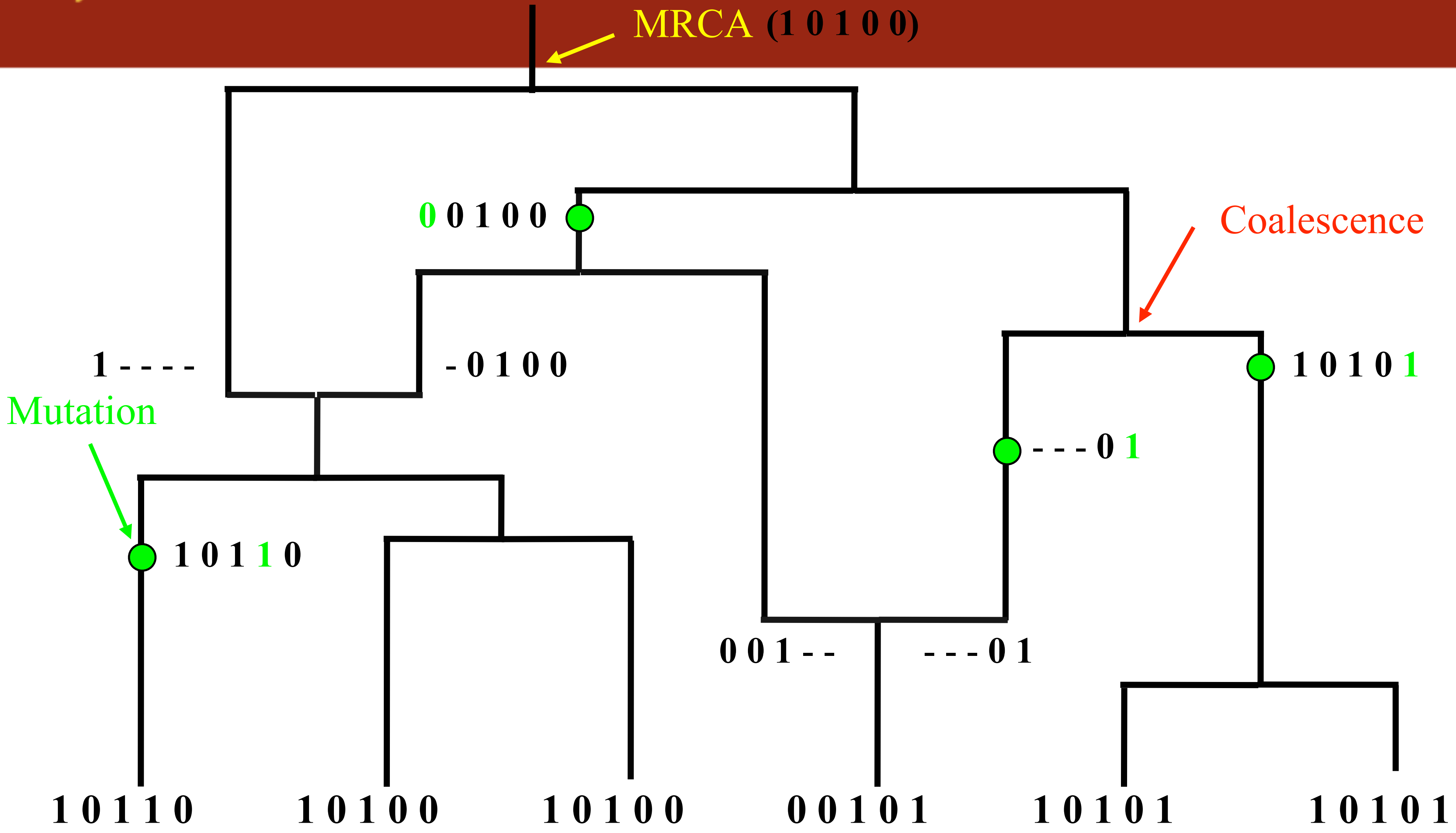
As we look back in time three events may occur:

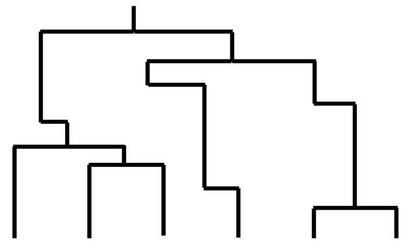
i. Two lines of ancestry will **coalesce** to form a single line of ancestry, with prob. $(k-1)/(k-1+\theta+\rho)$ where there are currently k lines and $\theta/2$ represents the mutation rate. (Pick a random pair of lines)

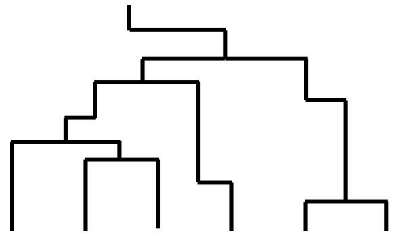
ii. A **mutation** will occur to a line of ancestry, changing the type of a gene, with prob. $\theta/(k-1+\theta+\rho)$. (Pick a random line)

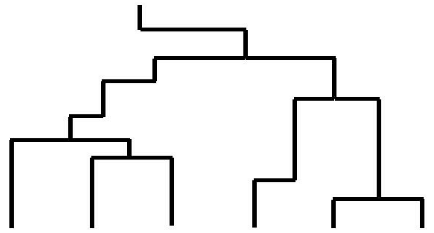
iii. A **recombination** will occur to a line, splitting it into two, with prob. $\varrho\ /(k-1+q+\rho)$. (Pick a random line)

The process continues until there is a single line of ancestry: the most recent common ancestor (MRCA) of the sample.




Tree for marker 1


Tree for markers 2 & 3


Tree for markers 4 & 5

Points of interest

- Not all mutations on the recombination graph impact the sample.
- Not all recombinations impact the sample.
- The space of possible graph topologies is (very!) infinite (c.f. the finite space of possible coalescent tree topologies).

Ancestral Processes with Recombination

- *Key observation: Each locus still follows a coalescent*
- Explicitly allows for the non-independence of multiple loci and use all data simultaneously.
- Recombination makes life much more difficult. Can wait a *long* time for the MRCA.

Can the coalescent produce human data?

- “Calibrating a coalescent simulation of human genome sequence variation” Schaffner, et al. Genome Research, 15:1576-1583, 2005.

Approximating the model:

Fast "Coalescent" Simulation

Goal

- A faster way of producing coalescent data for chromosomal-length regions (cf. existing methods such as Hudson's ms)

Why? – natural progression

slow

quick

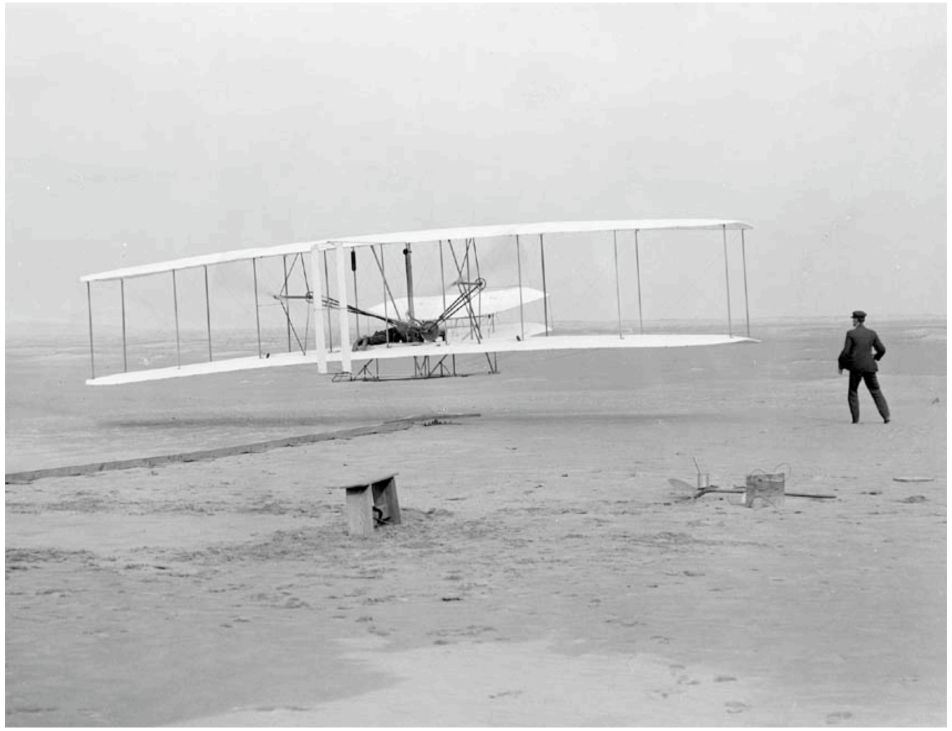
Why? – natural progression

slow

quick

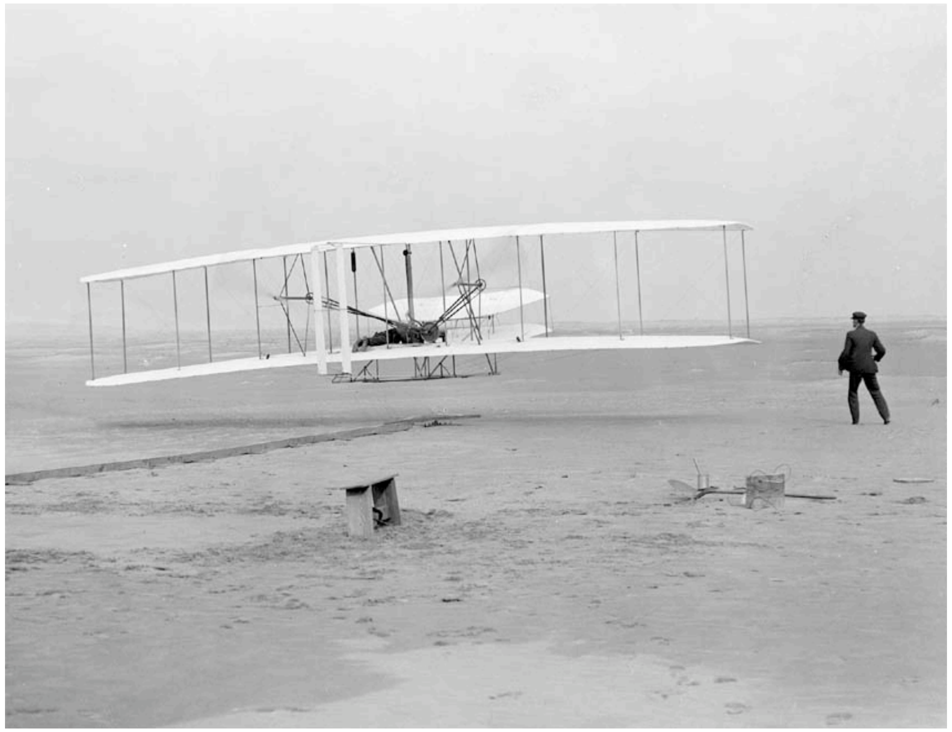
slow

quick



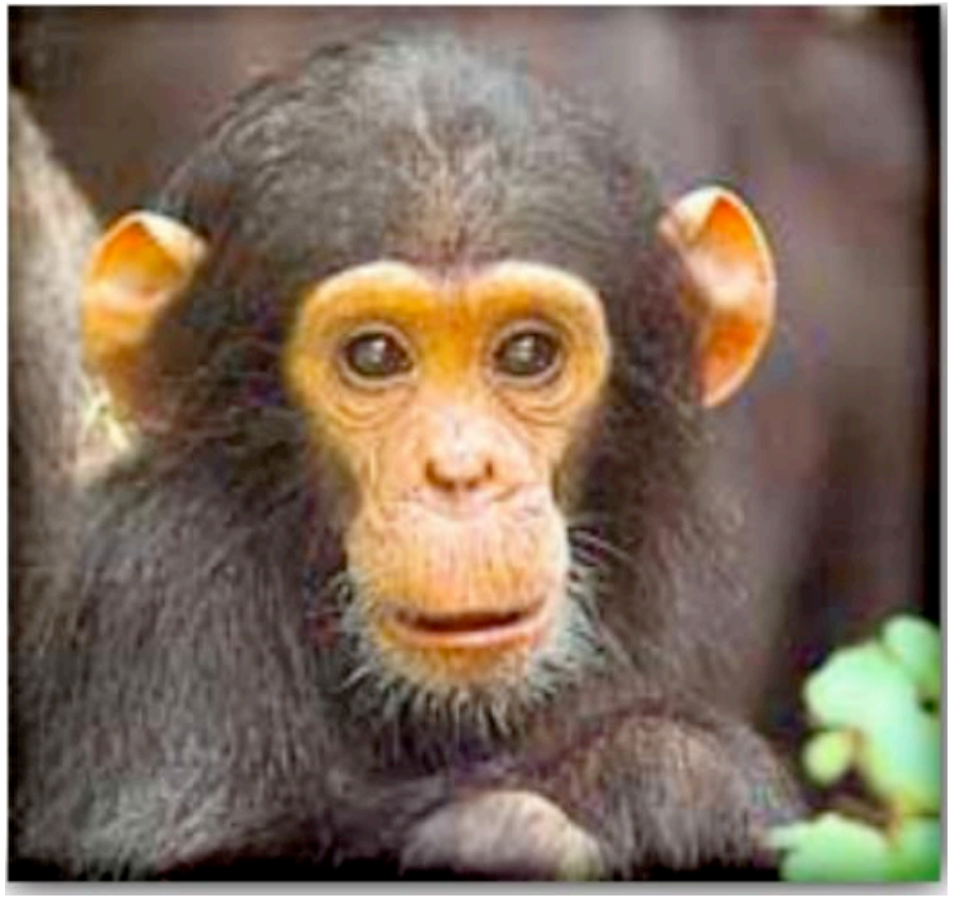
slow

quick

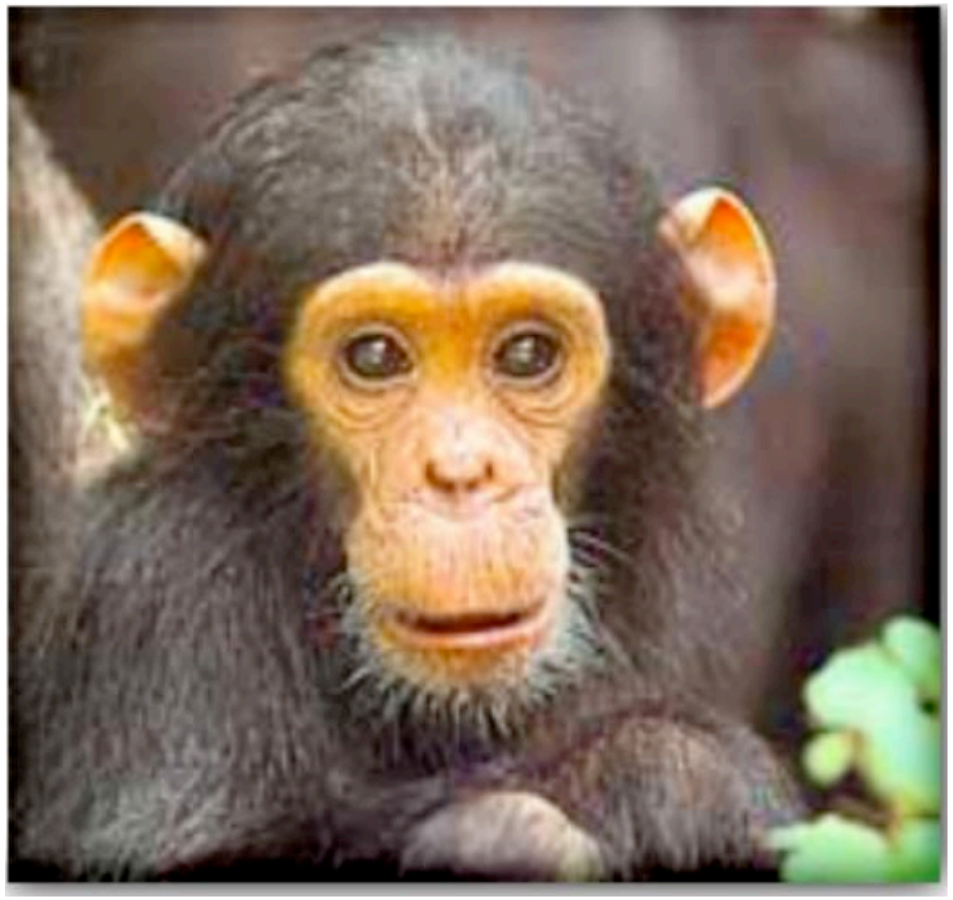


slow

quick

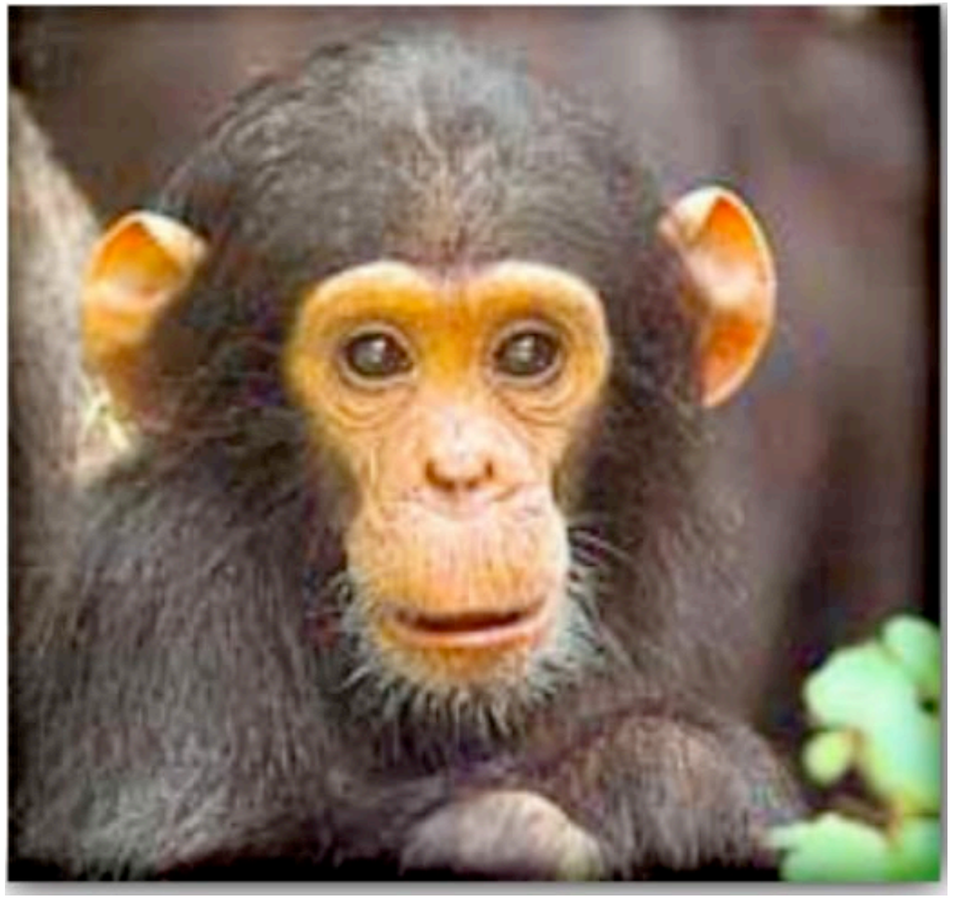


slow



quick

slow



quick

slow

quick

Why? - Growth of genome-wide data

- e.g. SNP-chips
- New analysis methodologies being developed. Need to test them somehow.
  - Usual strategy: simulate test data
  - Problem: traditional (coalescent) models too slow.
- Simulation-based analysis methods (Rejection algorithms, Importance Sampling, ‘no likelihoods’ MCMC - see part II)

Generating test data

- Real data + perturbation
  - e.g. bootstrap resampling
- Model + simulation
  - e.g. coalescent

Real data + perturbation

- Advantage – ‘model’ is correct.
  - Don’t know how the data got there, but it used the correct model.

- Disadvantage – subsequent perturbation adds noise. What do we end up with?

Model + simulation

- Advantage – Know what you are getting
- Disadvantage – May take a long while to get it + how accurate is the model?

Model-based approach

- Traditionally, many groups have used coalescent models
- Such models are slow for chromosomal-length regions

Full coalescent models are slow for chromosomal-length regions Run-times (secs) for ms (3 GB RAM)

Sample size	Length (Mb)	ms
1000	2	7.2
	5	62.6
	10	473.6
	20	6459.6
	50	-
	100	-
	200	-

Human chromosomes range from 50-200 Mbs

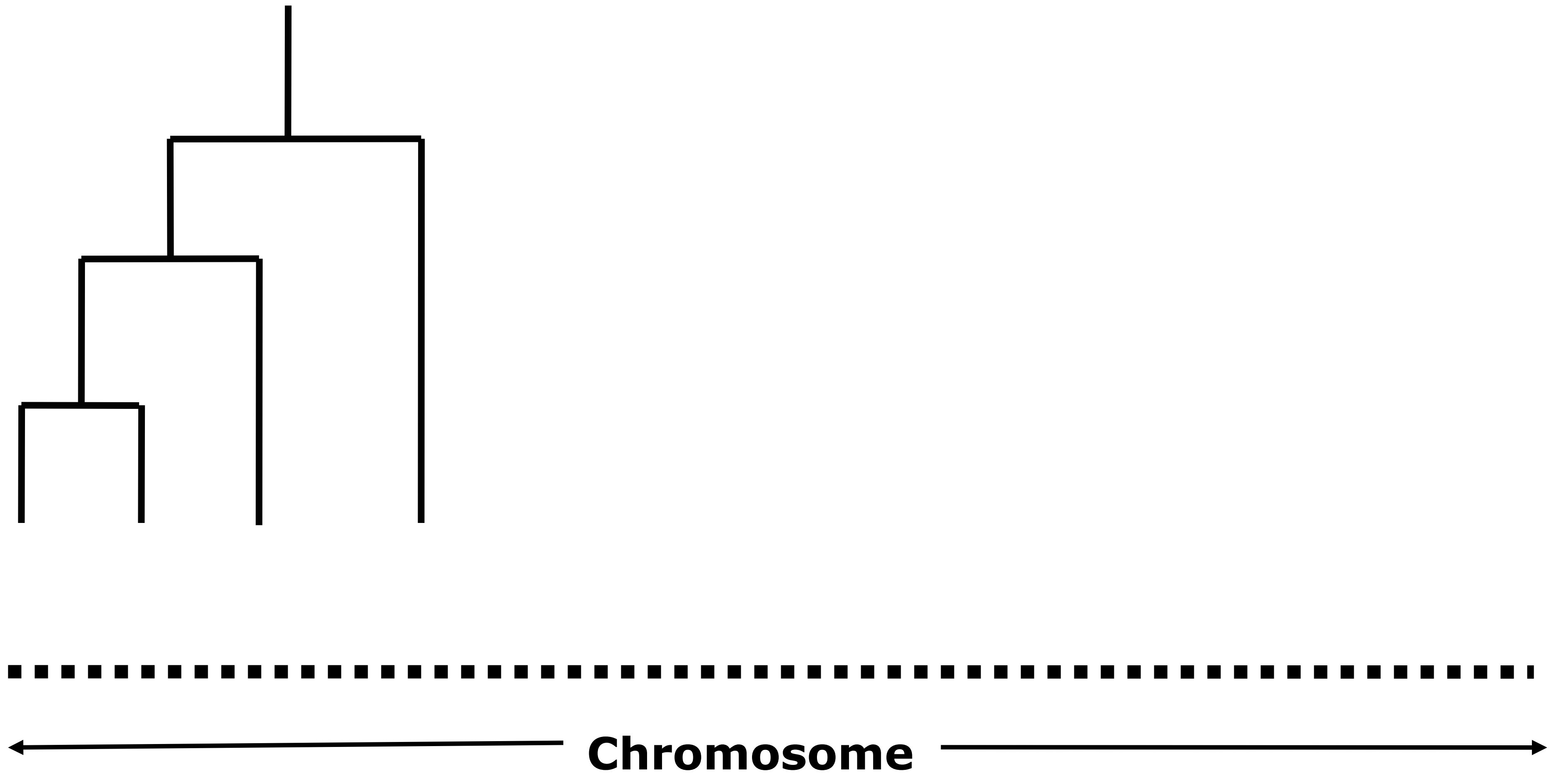
Run-times (secs) for ms (3 GB RAM)

Sample size	Length (Mb)	ms
4000	2	10.6
	5	-
	10	-
	20	-
	50	-
	100	-
	200	-

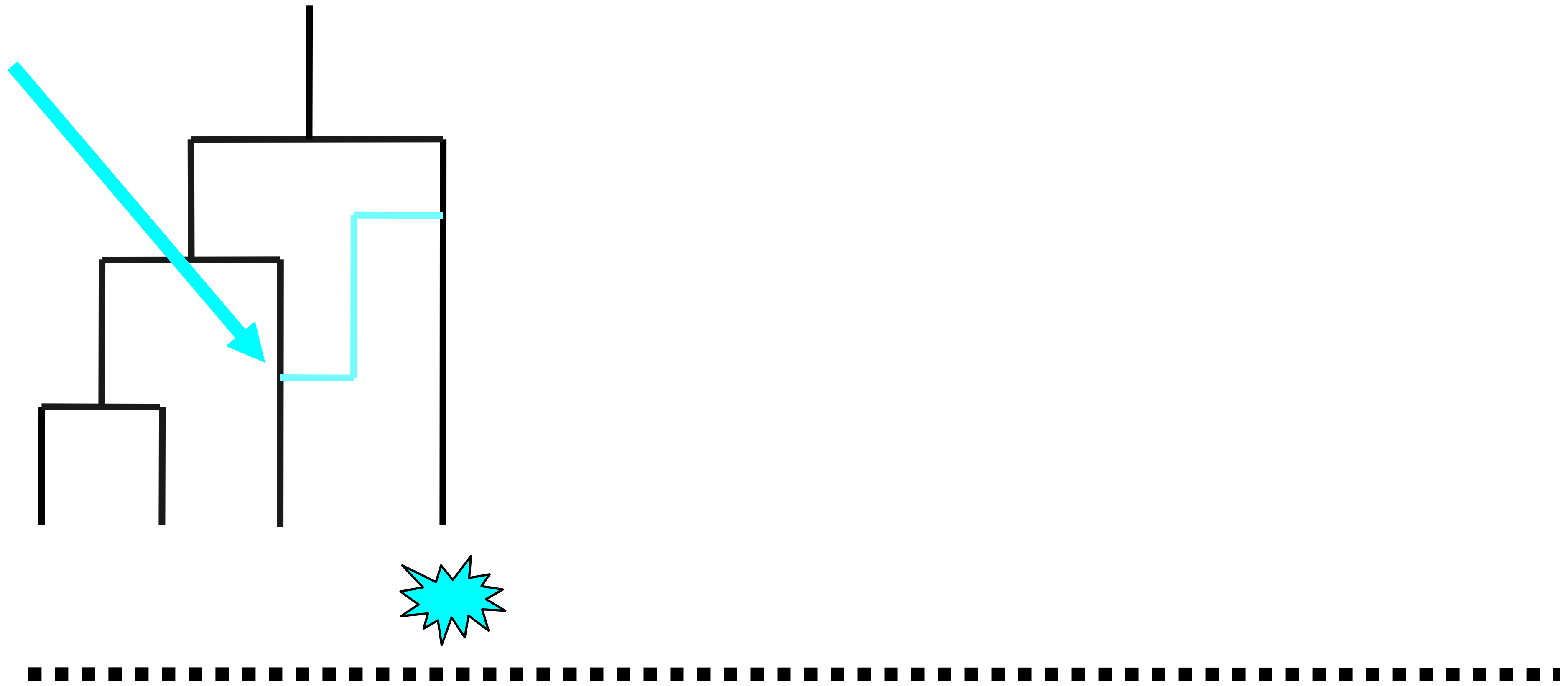
Find a faster way….How?

- Use an approximation to the coalescent
- Advantage - it will be faster
- Disadvantage – it's an approximation (to an approximation)

Wiuf and Hein “along the chromosome” algorithm

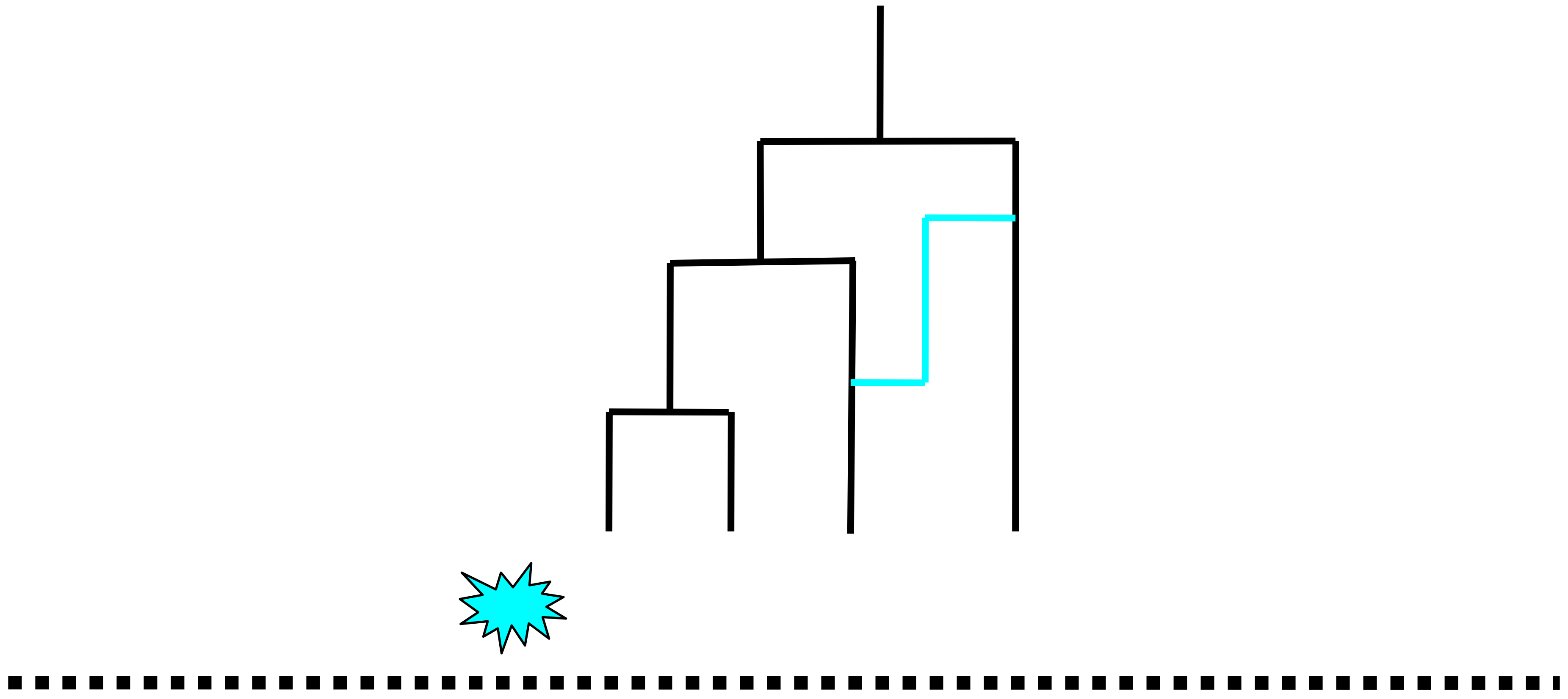


Wiuf and Hein “Along the



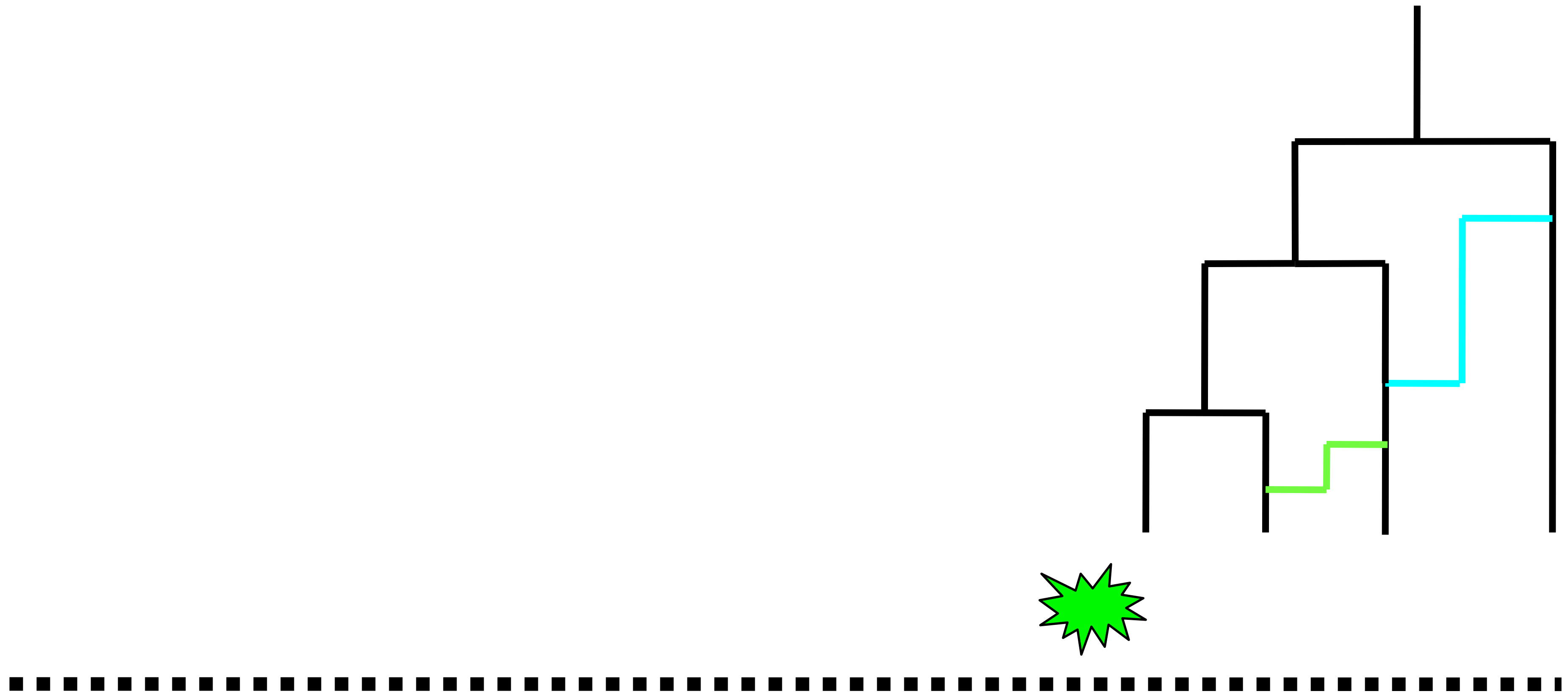
Chromosome

Wiuf and Hein “Along the Chromosome” algorithm



Chromosome

Wiuf and Hein “Along the Chromosome” algorithm

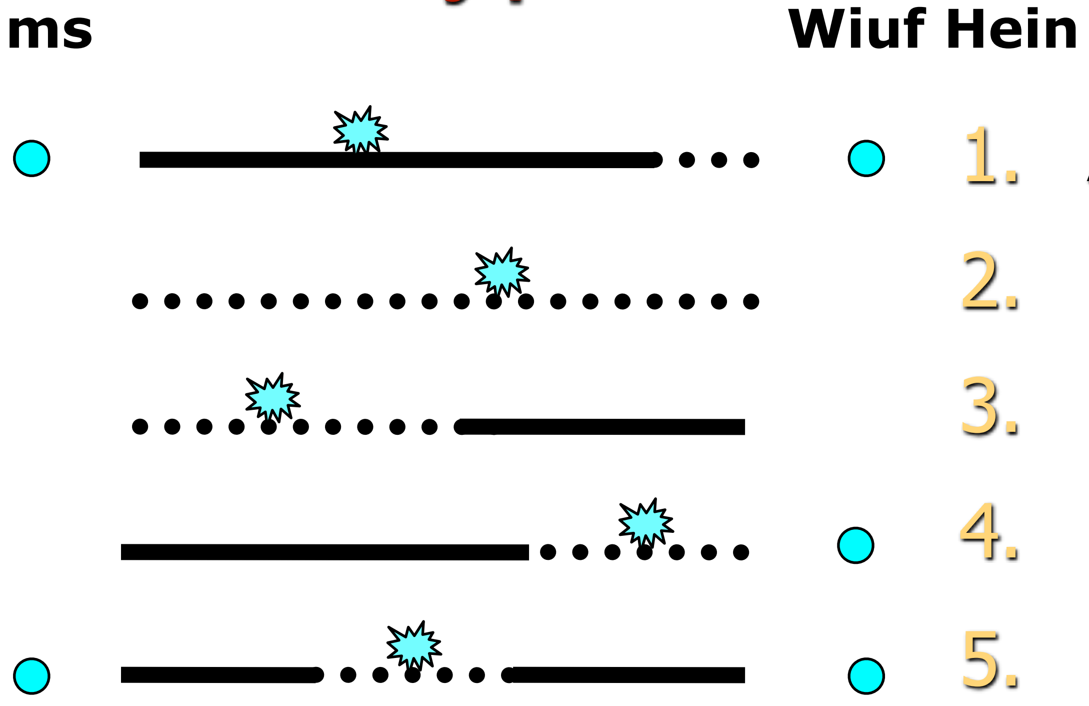


Chromosome

Comments

- Builds subset of ARG
- Slower than ms (larger subset)
  - Includes many recombinations in non-ancestral material
- Suggests a simplification

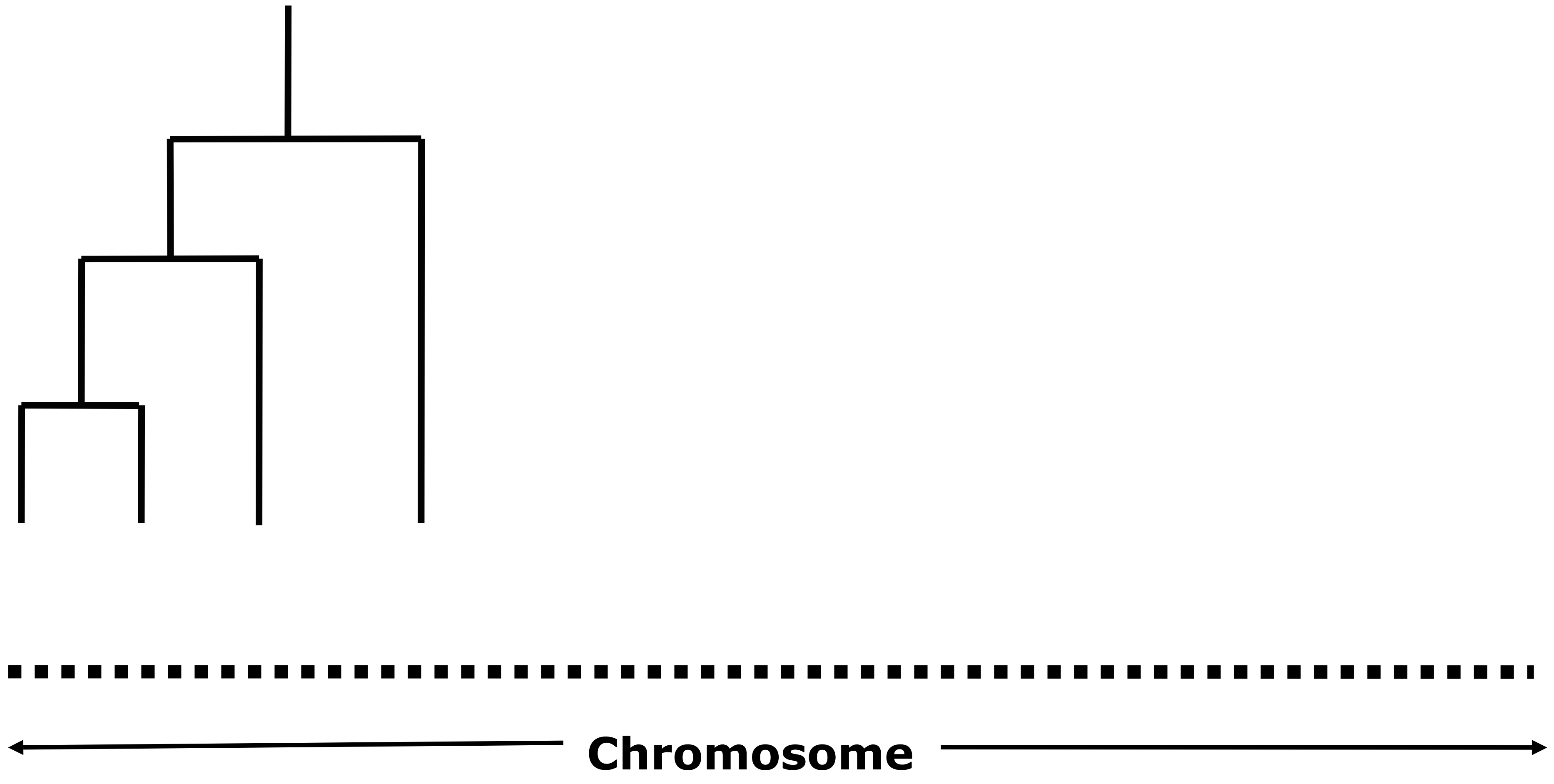
Types of recombination

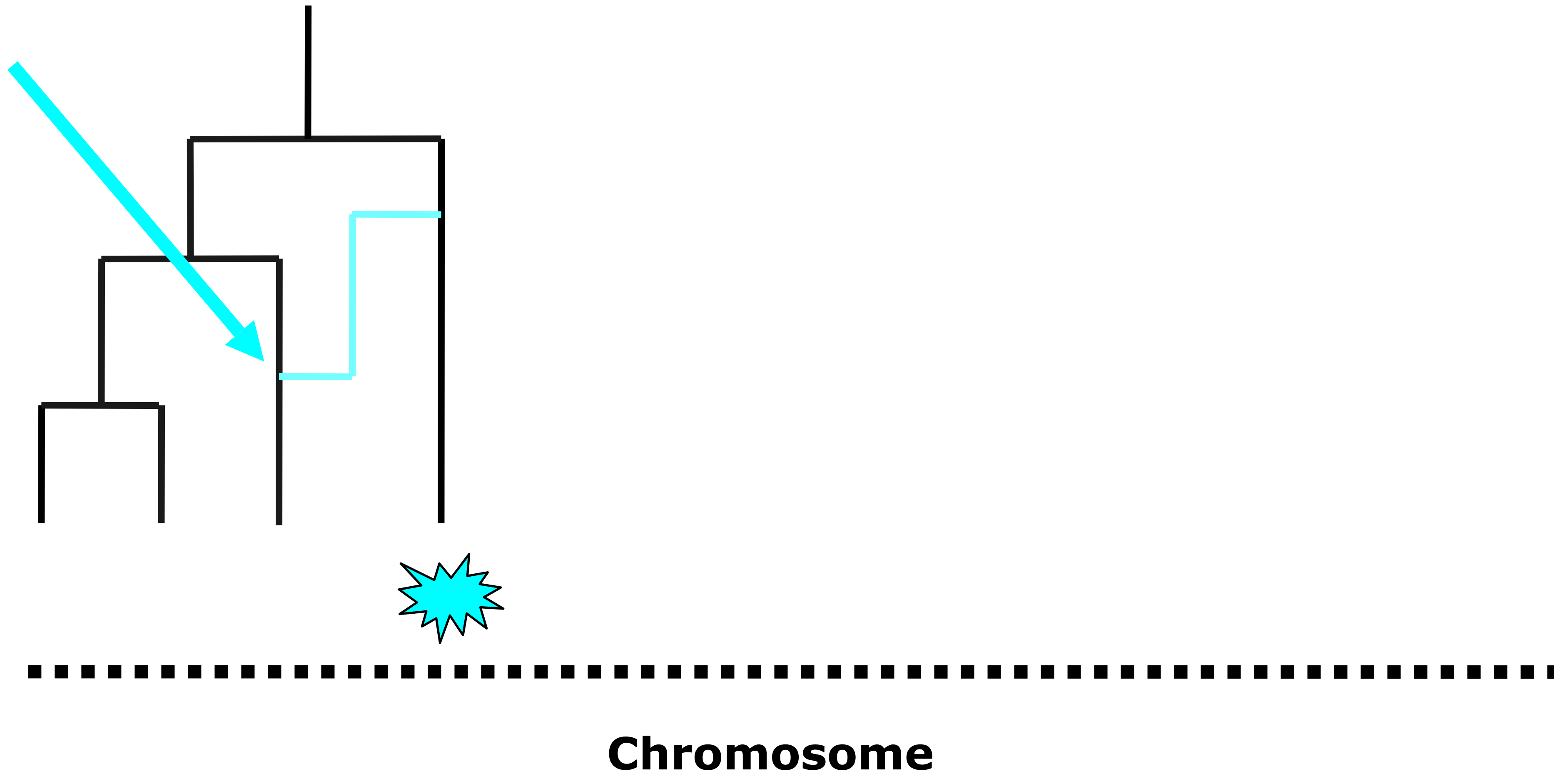


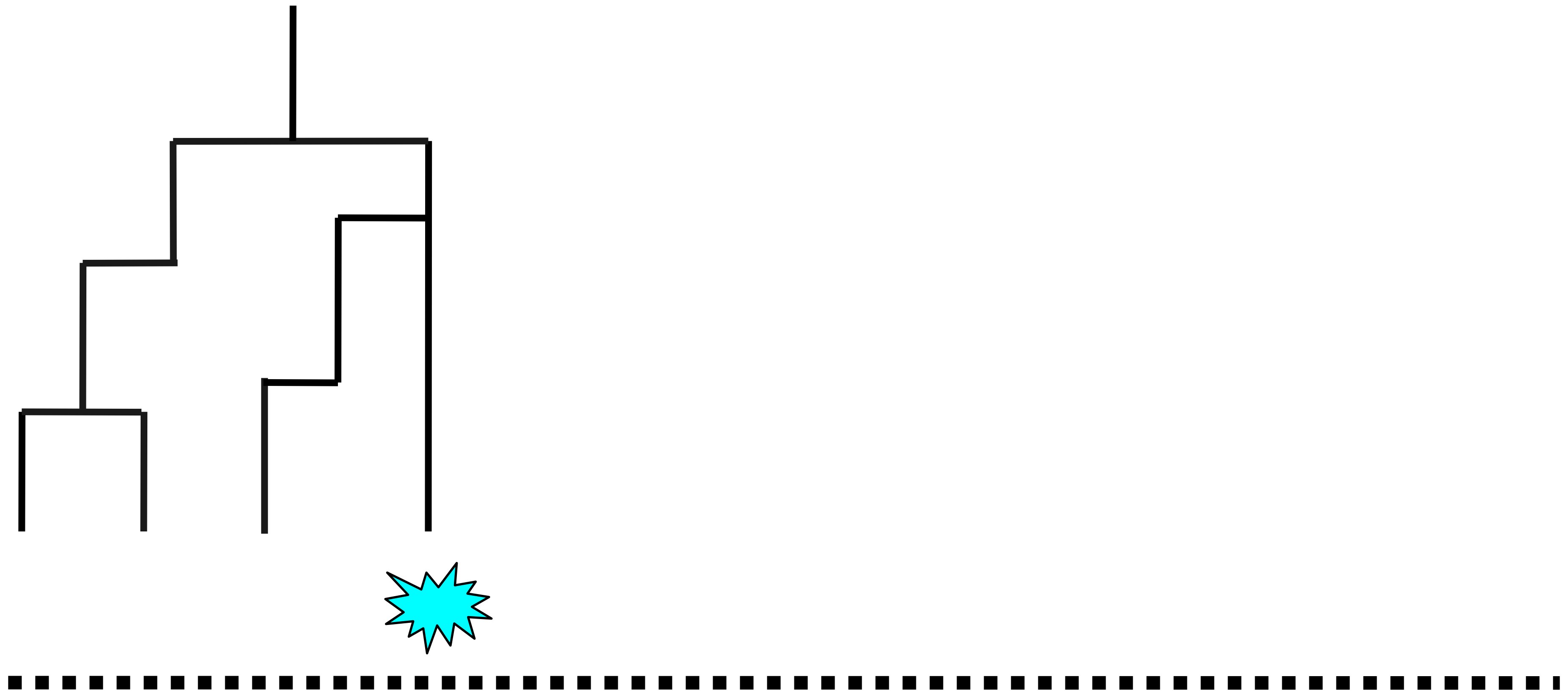
1. Ancestral material
2. Non-ancestral material
3. Non-ancestral material
4. Non-ancestral material
5. Non-ancestral material

▬▬▬	ancestral material
•••••••	non-ancestral material

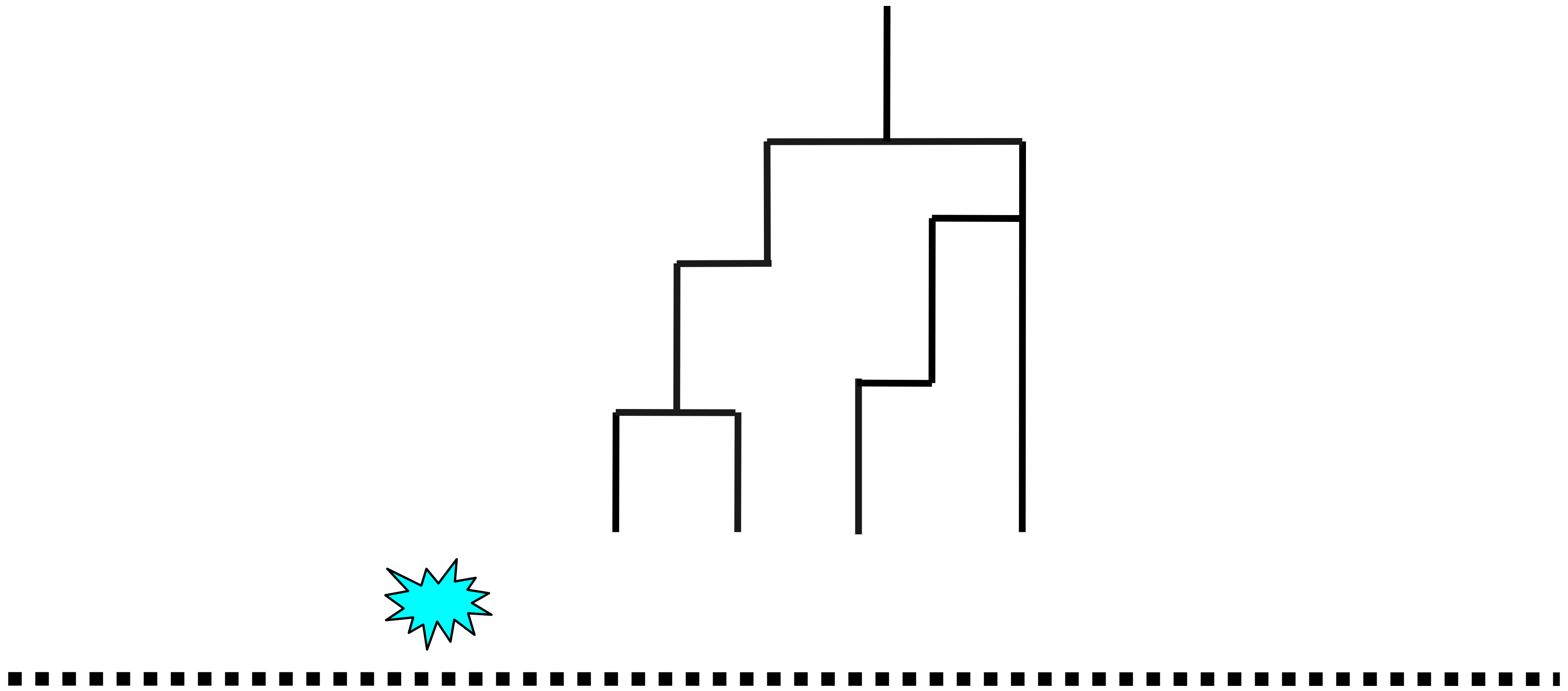
Sequential Markov Coalescent (McVean and Cardin 2005) (Marjoram and Wall 2006)


Chromosome

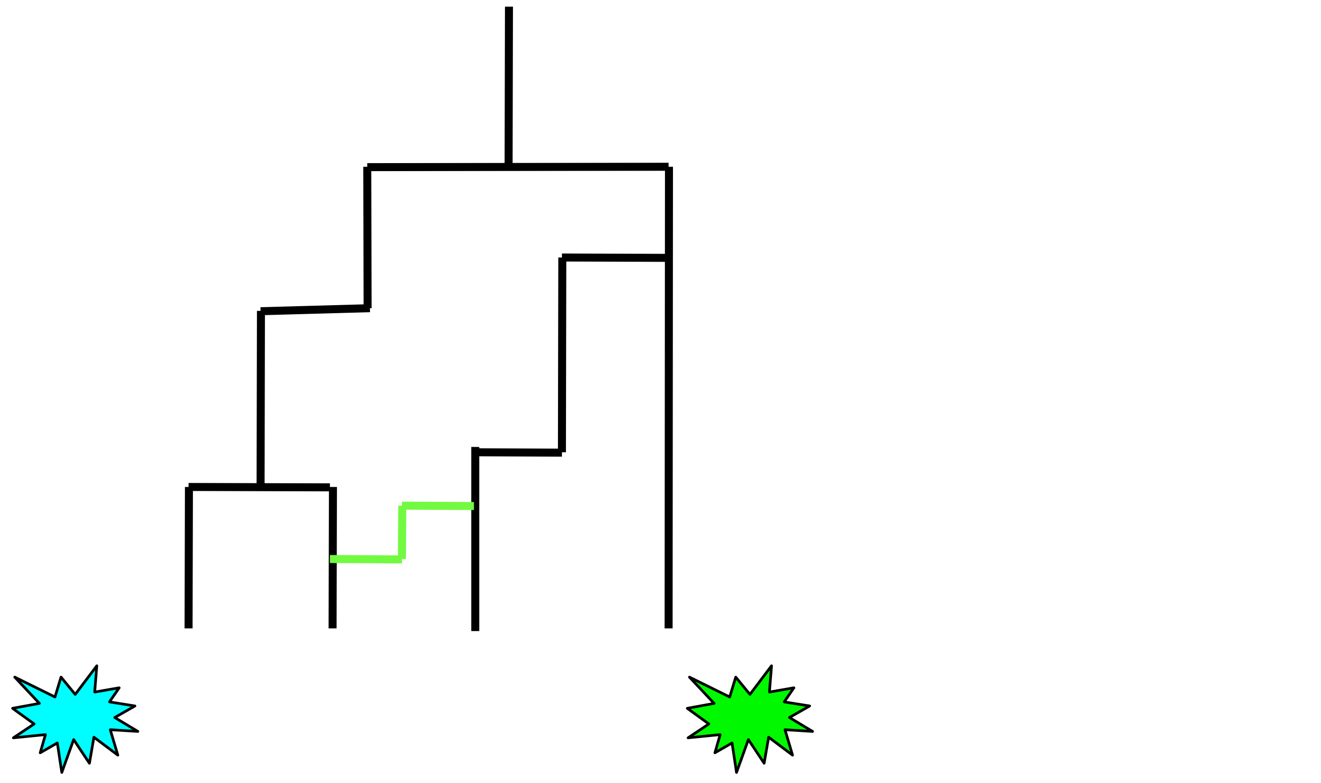

Chromosome



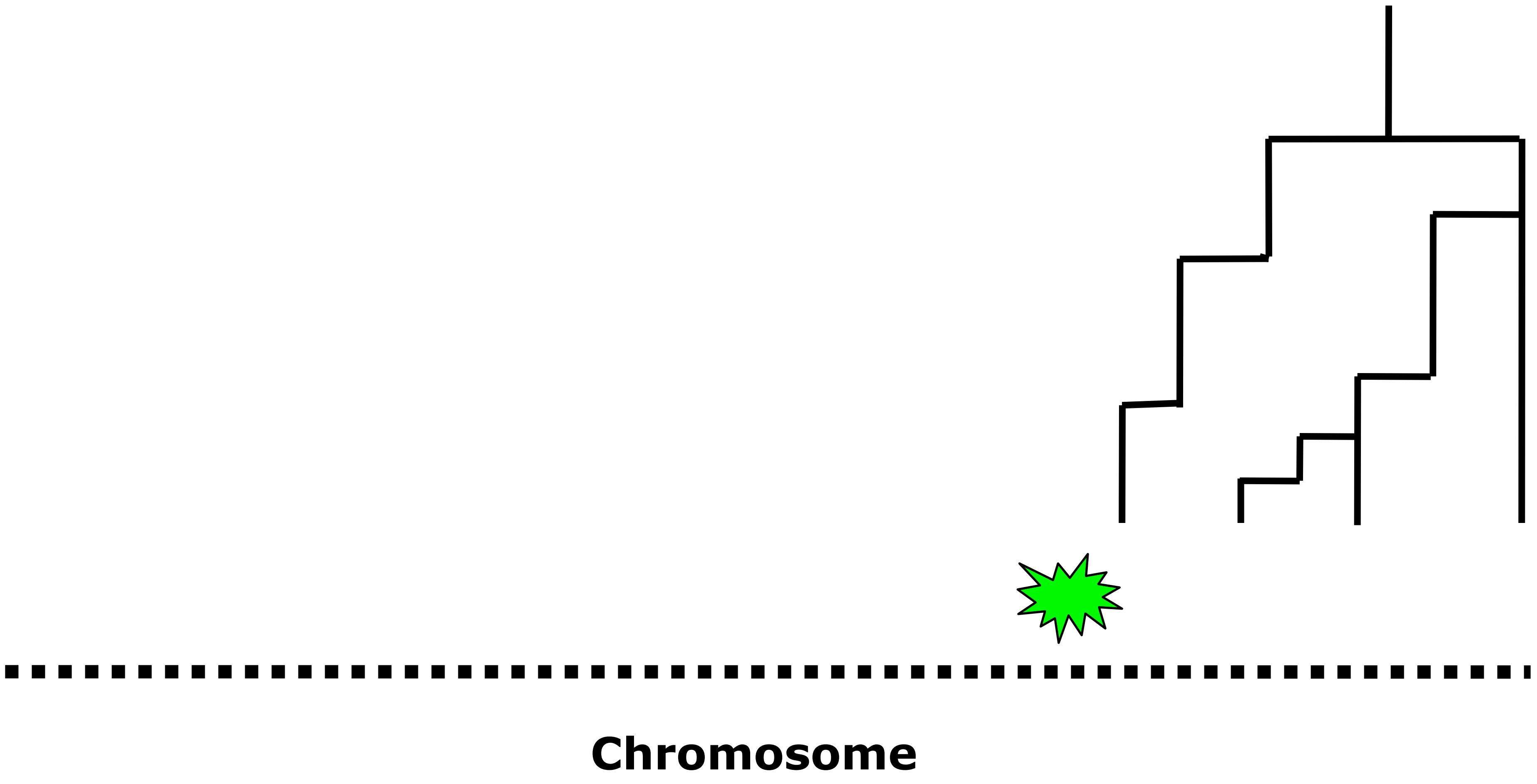
Chromosome



Chromosome



Chromosome


Chromosome

Outline of formal statement

- L(x): length of tree at x ϵ [0,1]
- Simulate y~Exp(L(x)ρ/2)
- If x+y<1
  - Start next tree at x+y by adding a recombination at a point chosen uniformly over the current tree
  - Add new line using usual coalescent prior
  - Delete old line
- Else
  - Stop

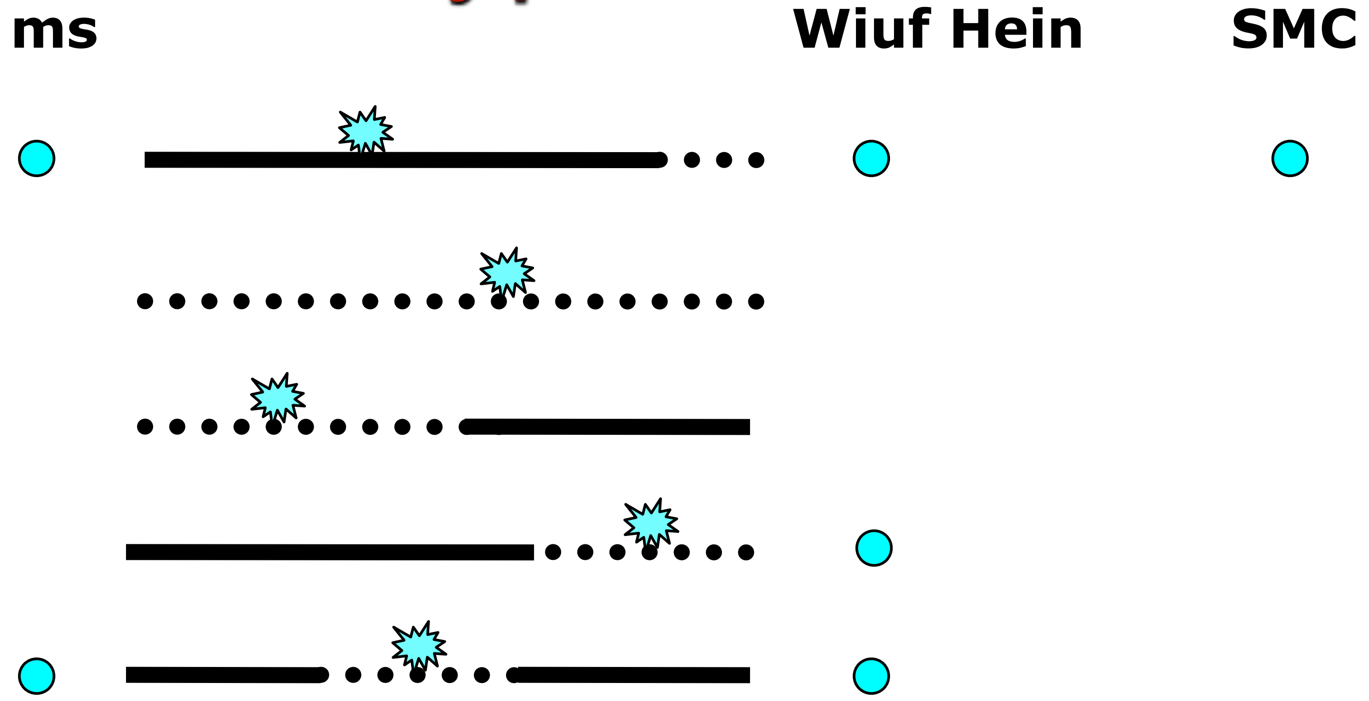
Run-times (secs) for ms (3 GB RAM)

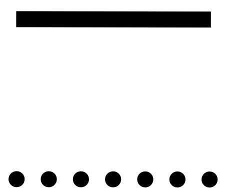
Sample size	Length (Mb)	ms	SMC
1000	2	7.2	0.9
	5	62.6	2.1
	10	473.6	4.3
	20	6459.6	8.3
	50	-	20.9
	100	-	41.6
	200	-	83.9

Run-times (secs) for ms (3 GB RAM)

Sample size	Length (Mb)	ms	SMC
4000	2	10.6	4.0
	5	-	10.4
	10	-	22.2
	20	-	40.7
	50	-	105.8
	100	-	201.5
	200	-	406.1

Types of recombination





ancestral material

non-ancestral material

Generalizations

- Now includes:
  - Variation in population size
  - Population structure
  - Gene conversion
  - Everything that ms does
- MACS (Chen et al. 2009)

- Agreement between MACS and ms is very good.
- When you can use ms, you should do so.
- For long regions, MACS provides a very close approximation to an exact answer that is otherwise unobtainable

- **Part II - Approximating the analysis**

'Vanilla' Rejection method

1. Generate θ from prior π.
2. Accept θ with probability $P(D|\theta)$. [Acceptance rate]
3. Return to 1.

- Set of accepted θ's forms empirical estimate of $f(\theta|D)$

- If upper bound, c, for $P(D|\theta)$ is known replace 2. with

  2′. Accept θ with probability $P(D|\theta)/c$.

- In general, $P(D|\theta)$ cannot be computed, so.

Alternate rejection method

1. Generate θ from π.
2. Simulate D′ using θ.
3. Accept θ if D′=D.
4. Return to 1.

Prob. may be v. small
Method then very inefficient

- (Likelihood estimation - Diggle and Gratton, J.R.S.S. B, 46:193-227, 1984.)

Rejection method - (approximate Bayesian computation)

- Suppose we have a good summary statistic S.

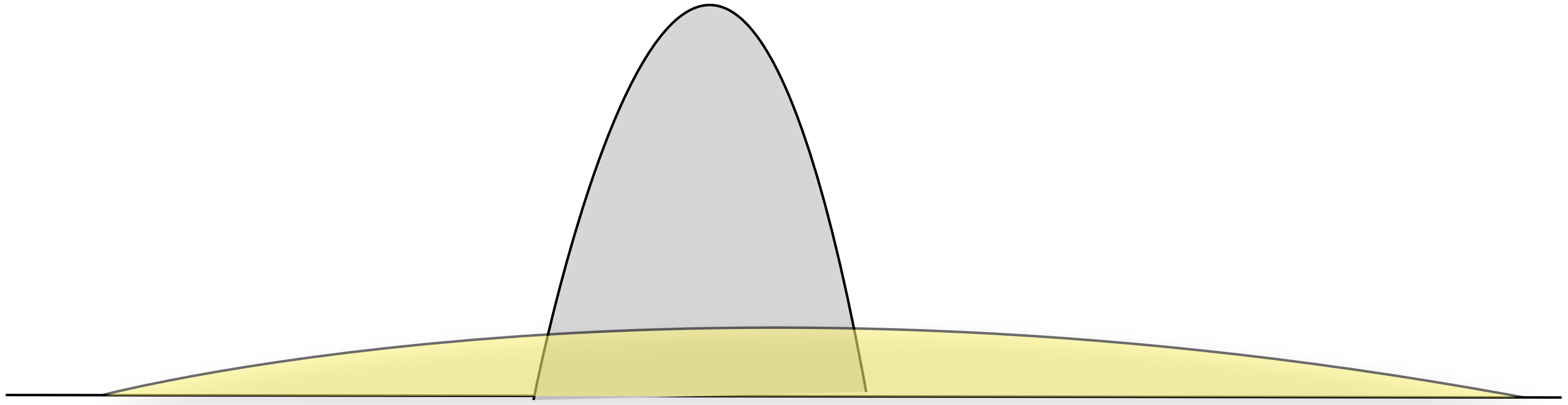
1. Generate θ from п.
2. Simulate D′ using θ, and calculate S′.
3. Accept θ if S′=S.
4. Return to 1.

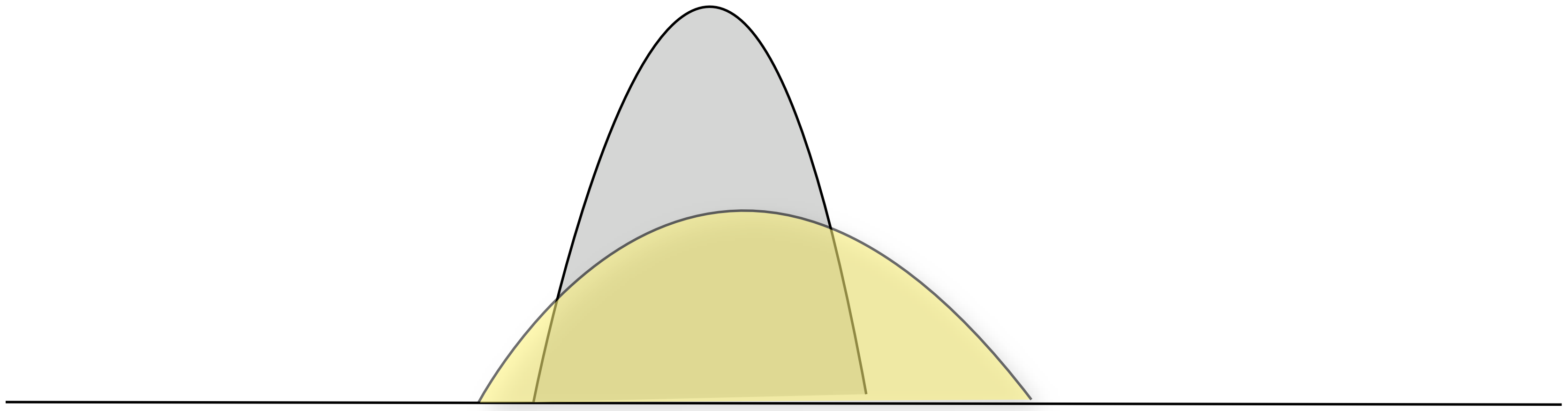
- Result: f(θ|S) [rather than f(θ|D)].

- Best case scenario: S is sufficient

- We know what are getting: $f(\theta | S)$
- If no sufficient statistic(s) S:
  - How to choose S?
  - How close is $f(\theta | S)$ to $f(\theta | D)$?
  - Lack of theoretical groundwork/guidance

Efficiency (c.f. Importance sampling)





MCMC - Metropolis-Hastings

1. If at θ, propose move to θ’ according to ‘transition kernel’ q(θ → θ’)

2. Calculate

$$h = \min\left\{1, \frac{P(D \mid \theta')\pi(\theta')q(\theta' \to \theta)}{P(D \mid \theta)\pi(\theta)q(\theta \to \theta')}\right\}$$

3. Move to θ’ with prob. h, else remain at θ

4. Return to 1.

Result: f(θ|D) ((Metropolis et al. 1953, Hastings 1970)

MCMC ‘without likelihoods’

1. If at θ, propose move to θ’ according to ‘transition kernel’ q(θ → θ’)

2. Generate data D’ using θ’

3. If D’=D go to 4; else stay at θ and go to 1

4. Calculate

$$h = \min\left\{1, \frac{\pi(\theta')q(\theta' \to \theta)}{\pi(\theta)q(\theta \to \theta')}\right\}$$

5. Move to θ’ with prob. h, else remain at θ

6. Return to 1.

Result: f(θ|D)

MCMC ‘without likelihoods’

1. If at θ, propose move to θ’ according to ‘transition kernel’ q(θ → θ’)
2. Generate data D’ using θ’, calculate S’
3. If S’=S go to 4.; else stay at θ and go to 1
4. Calculate

$$h = \min\left\{1, \frac{\pi(\theta')q(\theta' \to \theta)}{\pi(\theta)q(\theta \to \theta')}\right\}$$

5. Move to θ’ with prob. h, else remain at θ
6. Return to 1.

Result: f(θ|S)

How to choose statistics (Paul Joyce)

- Can't just include 'any and all' statistics (efficiency), so...
- Idea motivated by the concept of sufficient statistics.
- If S_1 is sufficient for θ, then:
  - $P(\theta|S_1)=P(\theta|D)$;
  - $P(\theta|S_1,S_2)=P(\theta|S_1)$ for any S_2 (but will be less efficient – lower acceptance rate)

Definition A set of statistics $S_1, S_2, \cdots, S_{k-1}$ are $\epsilon-$sufficient relative to a statistic X if

$$\sup_\theta \ln P(X|S_1, S_2, \cdots, S_{k-1}, \theta) - \inf_\theta \ln P(X|S_1, S_2, \cdots, S_{k-1}, \theta) \leq \epsilon$$

Definition The score of S_k relative to $S_1, S_2, \cdots, S_{k-1}$ is defined as follows.

$$\delta_k = \sup_\theta \ln P(S_k|S_1, S_2, \cdots, S_{k-1}, \theta) - \inf_\theta \ln P(S_k|S_1, S_2, \cdots, S_{k-1}, \theta).$$

"Add statistics until score for next statistic drops below Δ"

Procedure

- Suppose a data-set D and a set of possible statistics $S_1,\ldots,S_M$
- For i=1,...,N (N, very large):
  - Sample θ_i from prior $\pi()$
  - Simulate data D_i
  - Calculate $S_{1,i},S_{2,i},\ldots,S_{M,i}$
- Start with no statistics in the rejection method

Algorithm (applied to rejection method)

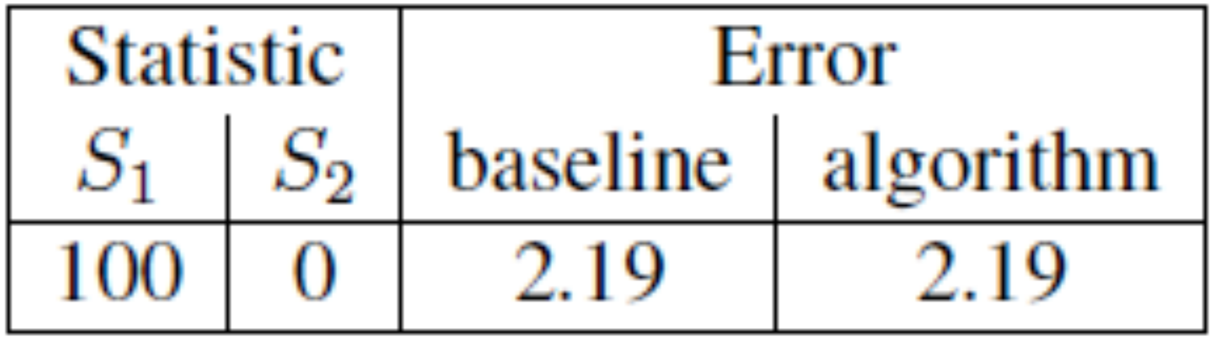
- Existing posterior, F_{k-1}, using S_1, S_2, ... , S_{k-1}
- Calculate posterior, F_k, after edition of randomly chosen currently unused stat S_k
- If $||F_k-F_{k-1}||$ “sufficiently large” add S_k
- Else do not include S_K
- If S_K added, try to remove S_1,...,S_{k-1}
- Repeat until no statistic can be added
- Stochastic noise is an issue

Example 1: Ewens Sampling formula

- Describes distribution of types in ‘infinite sites’ model
- Mutation parameter θ
- Number of types, S, is sufficient for θ
- Use sample size N=50

Statistics:

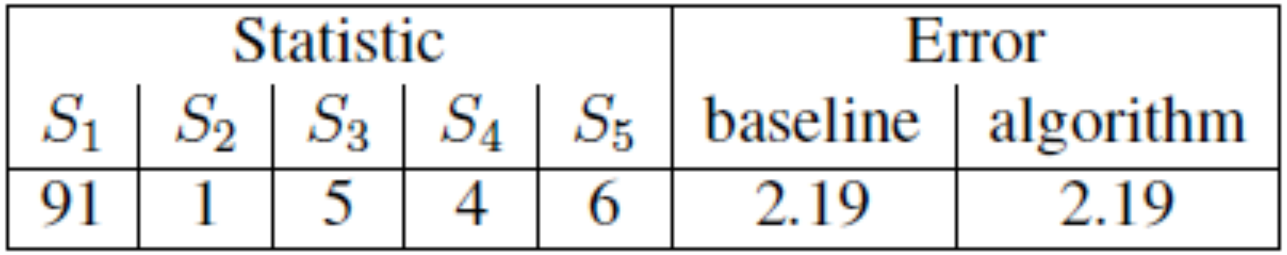
- S_1: S (the number of types)
- S_2: p (a random number ~ U[0,25])
- Use 5 million data sets and employ algorithm
- Analyze 100 datasets



Statistic		Error	
S_1	S_2	baseline	algorithm
100	0	2.19	2.19

More statistics:

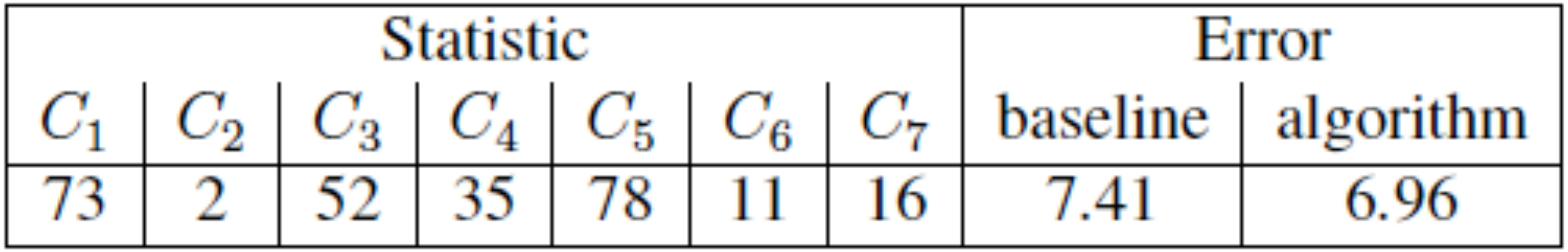
- S_1: S (the number of types)
- S_2: p (a random number ~ U[0,25])
- S_3: 50x Homozygosity
- S_4: 25x frequency of commonest type
- S_5: Number of singleton types



Statistic					Error	
S_1	S_2	S_3	S_4	S_5	baseline	algorithm
91	1	5	4	6	2.19	2.19

Example 3: coalescent, estimate ρ

- C_1: #mutations
- C_2: U[0,25]
- C_3: mean # pairwise differences
- C_4: 25x mean pairwise LD between ‘nearby’ loci
- C_5: #haplotypes
- C_6: #copies of commonest haplotype
- C_7: #singleton haplotypes



Statistic							Error	
C_1	C_2	C_3	C_4	C_5	C_6	C_7	baseline	algorithm
73	2	52	35	78	11	16	7.41	6.96

Approach 2 - Genetic algorithms

- A population of ‘algorithms’
- Each algorithm has a ‘fitness’
- Evolve through discrete generations
- Algorithms reproduce according to their fitness
- Subject to mutation and recombination

Trivial example

- Algorithm = vector of 8 binary numbers
- Fitness = # of 1s
  - e.g. 00010010 --> fitness=2
  - e.g. 11010110 --> fitness=6
- Mutation: point mutation (flip a bit)
- Recombination: choose a breakpoint and concatenate two parents
  - 110100100 + 000010111
  - > 110010111

Results - time to find fittest algorithm

- Using vectors of length 20, population size=100, p(mutate)=0.001/bit
- Mutation only: 609 generations

Results - time to find fittest algorithm

- Using vectors of length 20, population size=100, p(mutate)=0.001/bit
- Mutation only: 609 generations
- Mutation + recombination (prob=0.7): 75 generations

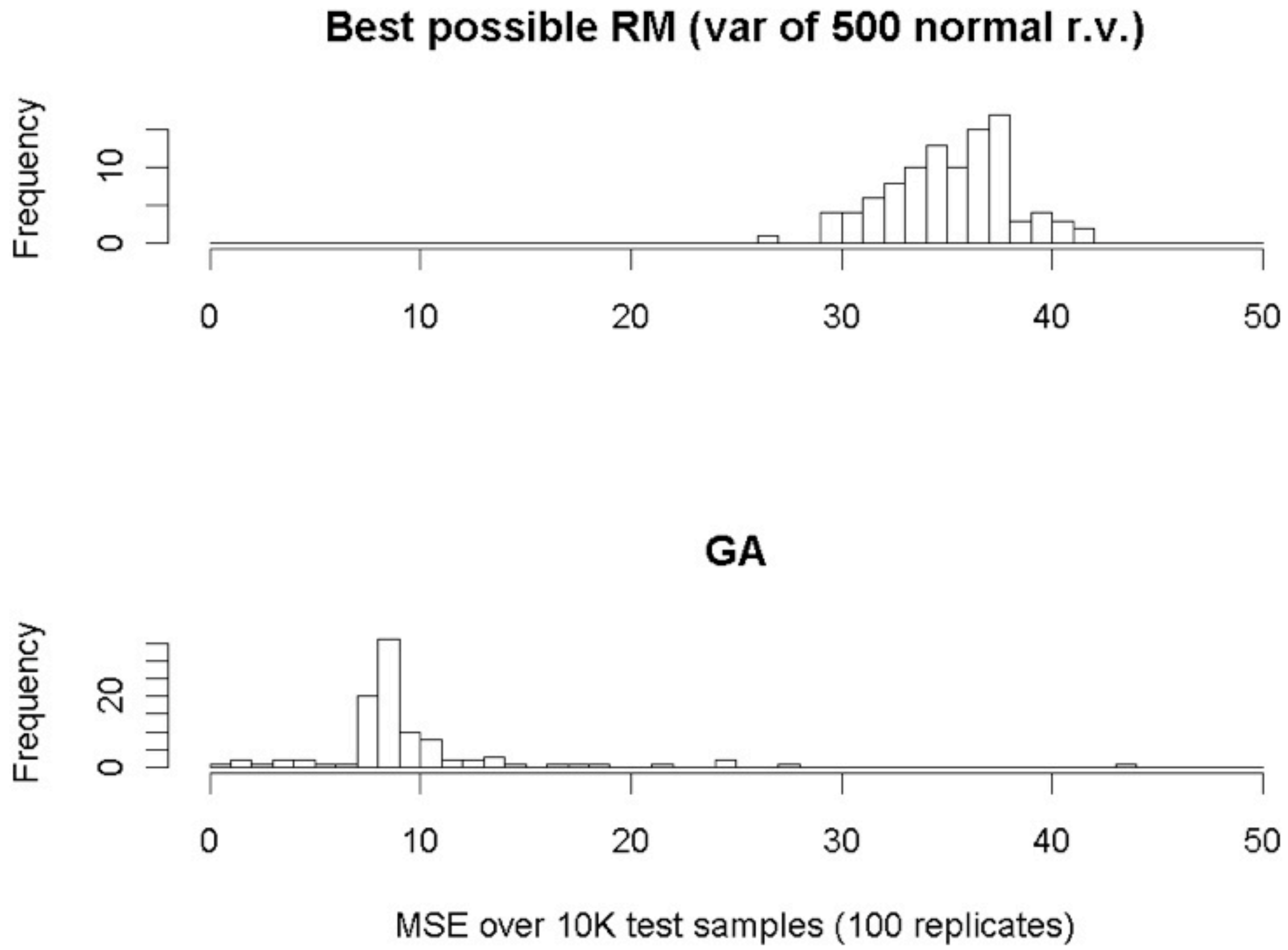
Back to rejection methods

- Want to pick arbitrary linear combination of summary statistics (S_1,....,S_n) that captures the information about θ
- Algorithm is now a vector of coefficients
  - e.g.

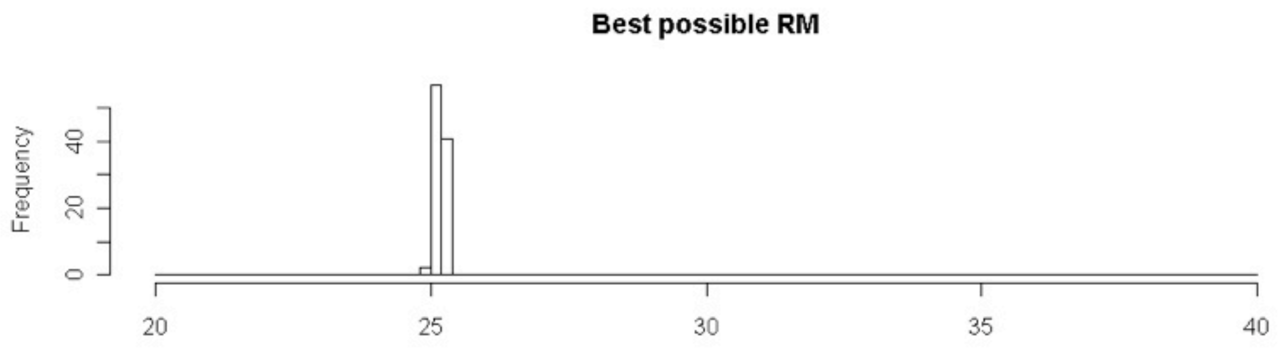
1.3	-5	0.01	16	-0.2
S_1	S_2	S_3	S_4	S_5

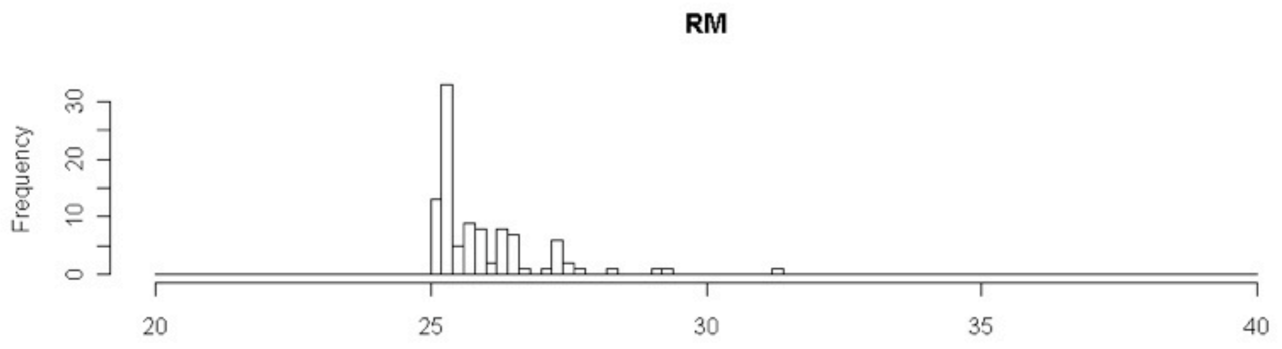
- Create 100 test data sets $D_1, \ldots, D_{100}$ sampling from prior θ
- Create pool of 5M (say) data sets, sampling θ from prior, to use as simulated data in rejection method
- For each algorithm, j, run rejection method for each D_i, calculate mean of posterior for θ_i
- Fitness is 1/(mean square error)
- Evolve for 50 generations
- Test final fittest GA on new set of 100 data sets.

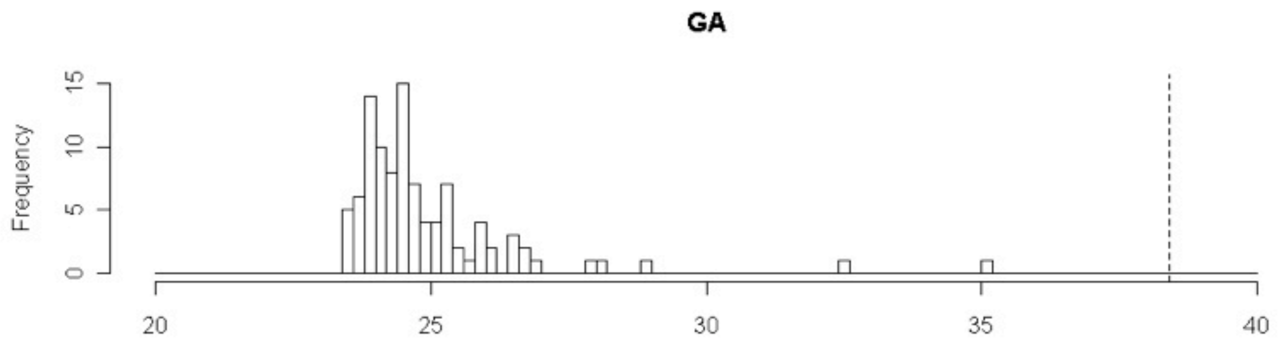
Estimating Normal variance



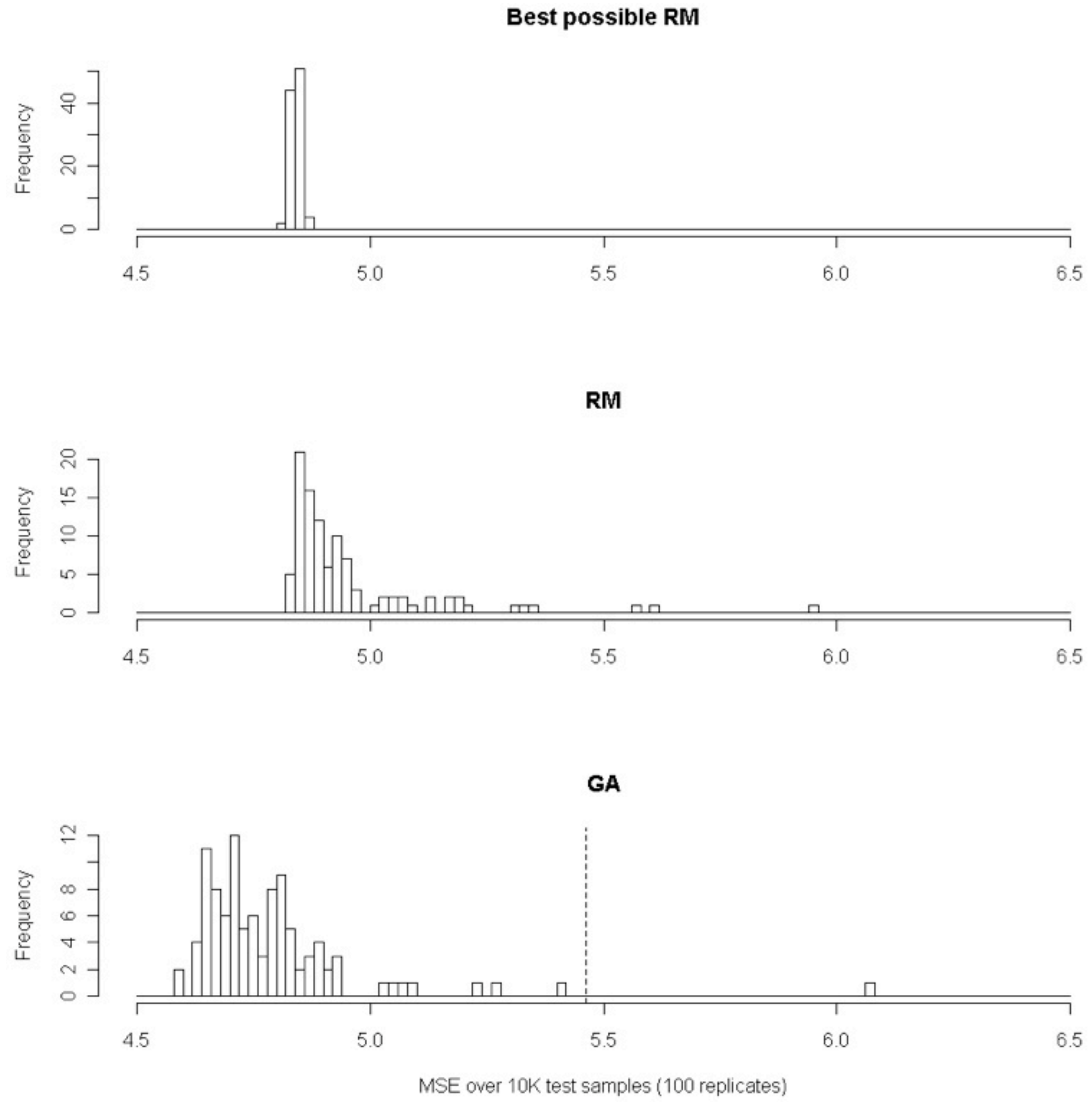
Estimating mutation rate







Estimating recombination rate



General comments

- Approximate methods allow analysis in situations where exact analysis is intractable
- Choice of summary statistics is problematic
- Two methods, both choose statistics sensibly on test examples, the genetic algorithm also chooses weights
- There remains a worrying absence of theory to tell you how well you are doing [i.e. how close is $P(\theta|S)$ to $P(\theta|D)$?]

- **Refs (Part I):**
  - Recombination as a point process along sequences, Wiuf and Hein, *Theor. Pop. Biol*. 55:28-259, 1999.
  - Approximating the coalescent with recombination, McVean and Cardin, *Phil. Trans. R. Soc. B* 360:1387–1393, (2005).
  - Fast "Coalescent" Simulation. Marjoram and Wall. *BMC Genetics*, 7:16, 2006.
  - Fast and flexible simulation of DNA sequence data, Chen, Marjoram Wall, *Genome Research*, 19:136-142, 2009
  - MACS algorithm available at http://hsc.usc.edu/~garykche

- **Refs (Part II):**
  - Approximately sufficient statistics and Bayesian computation. Joyce & Marjoram. Stat Appl Genet Mol Biol. 2008; 7:Article26. 2008

Collaborators

- Jeff Wall, Gary Chen
- Simon Tavaré, Paul Joyce, Hsuan Jung

END

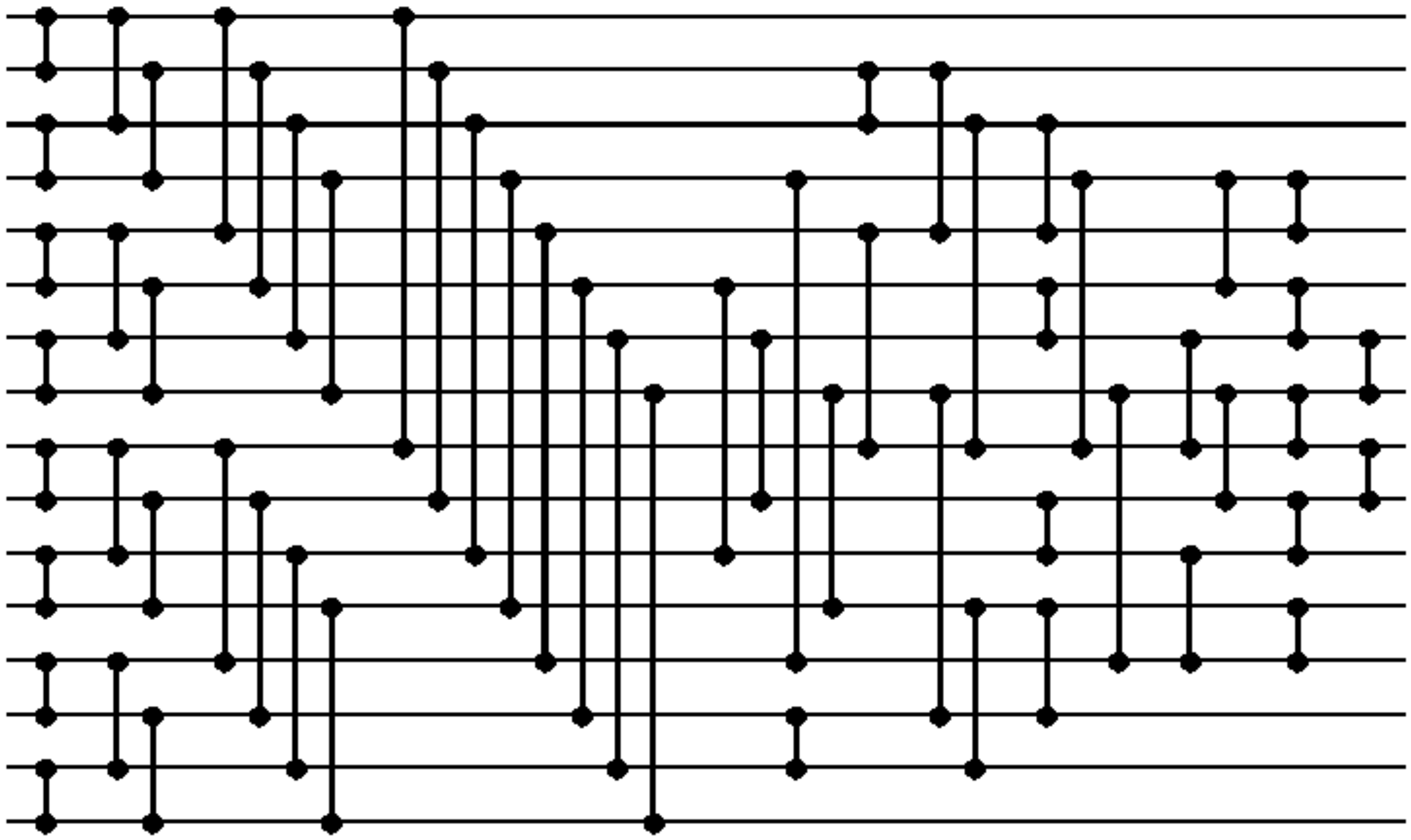
Other notes

- Generalizes to multiple variables
- Evolving the test data
  - keep the ‘hardest’ - sorting algorithms
  - keep the ‘easiest’ - noisy data?
- There is little theory
- Applications are seat-of-the-pants/ heuristic/intuitive

Pair-wise algorithms: history, n=16

- Let m = number of pairwise comparisons made
- 1962 - Bose and Nelson: m=65. Conjectured to be optimal.
- 1964 - Batcher, and Floyd & Knuth: m=63. Believed to be optimal.
- 1969 - Shapiro: m=62. Too smart to conjecture optimality......
- 1969 - Green: m=60. Remains the best solution.

Green’s 60 step sorter



http://www.cs.brandeis.edu/~hugues/graphics/green.gif

Genetic Algorithm

- Individuals encoded as ordered list of pairwise comparisons:

  5, 3, 6, 1, 2, 4.

  (1,4) (2,3) (3,6) (2,5) (3,5) (4,5)

Fitness

- Need a definition of fitness:
- For a given algorithm on a given sequence, count the number of pairs of adjacent numbers that are incorrectly ordered, N_p.
- $f = 1/(N_p+\varepsilon)$?
- Calculate a mean of f over a large number of test sequences of unordered numbers.

Result

- Population size = 512-1000000 individuals
- 5000 generations
- Best algorithm: length = 65

Pair-wise algorithms: history, n=16

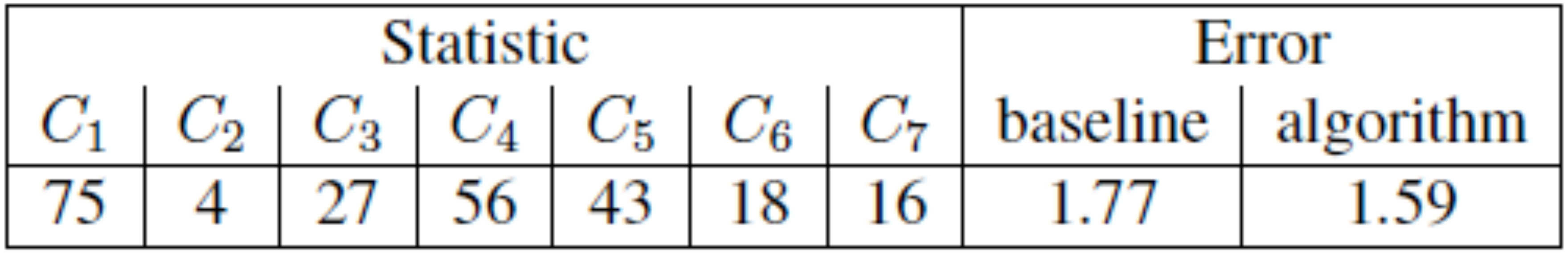
- Let m = number of pairwise comparisons made
- 1962 - Bose and Nelson: m=65. Conjectured to be optimal.
- 1964 - Batcher, and Floyd & Knuth: m=63, (see previous slide). Believed to be optimal.
- 1969 - Shapiro: m=62. Too smart to conjecture optimality......
- 1969 - Green: m=60.

Back to the drawing board....

- Ideas from host-parasite evolution
- View sorting algorithms as ‘hosts’
- View the test data sets of unordered numbers as ‘parasites’

Example 2: coalescent, estimate θ

- C_1: #mutations
- C_2: U[0,25]
- C_3: mean # pairwise differences
- C_4: 25x mean pairwise LD between ‘nearby’ loci
- C_5: #haplotypes
- C_6: #copies of commonest haplotype
- C_7: #singleton haplotypes



Statistic							Error	
C_1	C_2	C_3	C_4	C_5	C_6	C_7	baseline	algorithm
75	4	27	56	43	18	16	1.77	1.59

Coalescent - mutation rate

- S_0 = Number of types (nearly sufficient)
- S_1 = A random number (U[0,20])
- 50000 data sets

- After 10 generations of 20 algorithms:
  - fittest alg. is $79.0S_0 + 0.03S_1$

Coalescent mutation rate - more stats [SNP data]

- S_0 = Number of segregating sites (nearly sufficient)
- S_1 = A random number (U[0,20])
- S_2 = Number of ‘pairwise differences’
- S_3 = Mean pairwise linkage disequilibrium
- S_4 = Number of haplotypes

- fittest algorithm:
  - $0.8S_0 + 0.06S_1 + 6.0S_2 + 0.5S_3 + 28.0S_4$
- 5th fittest (very similar fitness)
  - $9.2S_0 + 0.07S_1 + 0.2S_2 + 0.3S_3 + 0.3S_4$

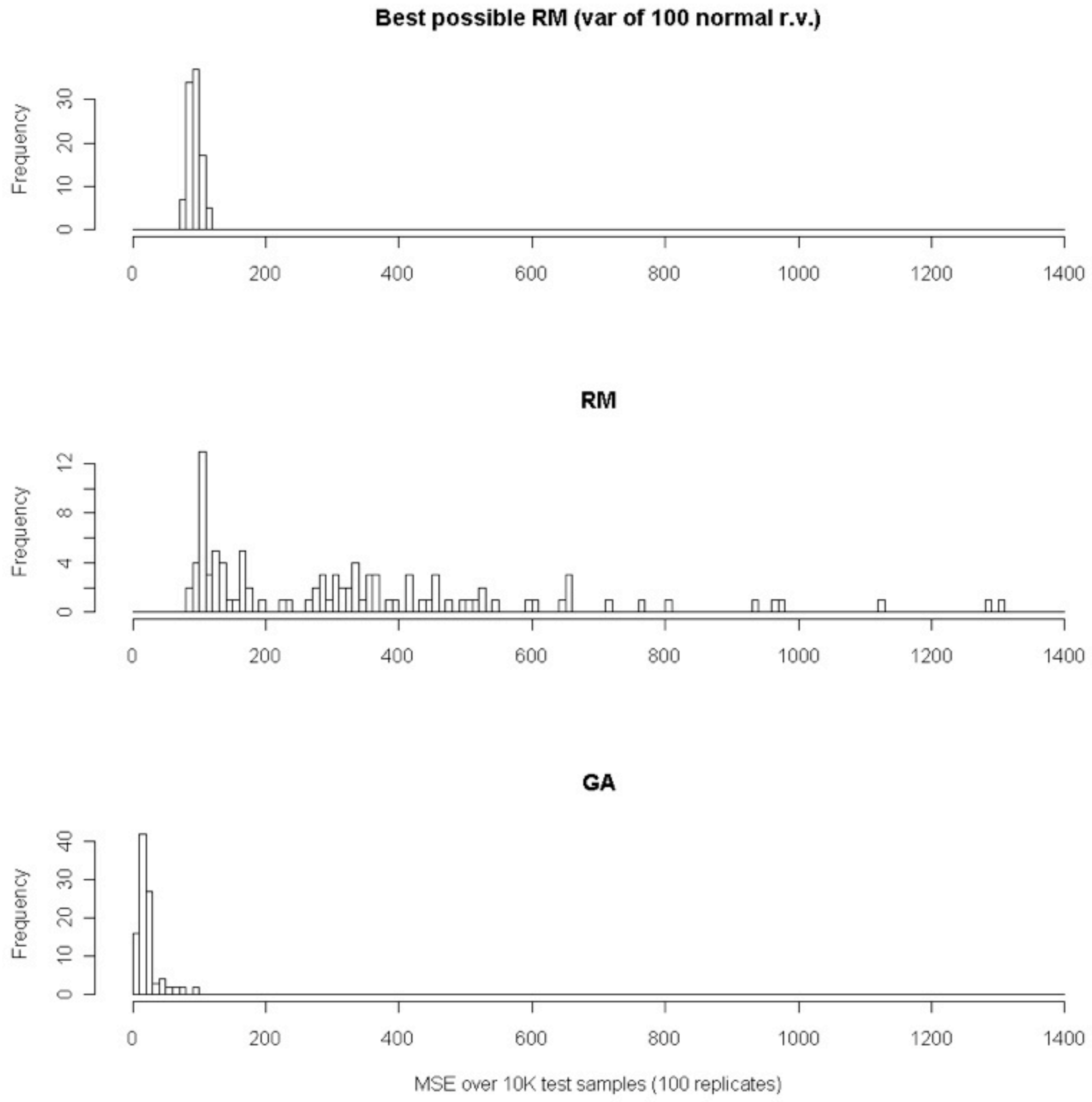
Same problem but with more data (250K vs. 50K)

- S_0 = Number of segregating sites (nearly sufficient)
- S_1 = A random number (U[0,20])
- S_2 = Number of ‘pairwise differences’
- S_3 = Mean pairwise linkage disequilibrium
- S_4 = Number of haplotypes

- fittest algorithm:
  - $34.1S_0 + 0.2S_1 + 0.6S_2 + 0.0S_3 + 95.8S_4$

- Define parasites that contain 10-20 test lists of numbers
- Have sorters and parasites evolve on a 2d grid
- Test an algorithm’s fitness using the nearest parasite
- Parasite fitness = % of lists that were not sorted correctly
- Evolve the parasites!
- Best solution: 61 comparisons.

Estimating Normal variance


Best possible RM (var of 100 normal r.v.)
Frequency
RM
Frequency
GA
Frequency
MSE over 10K test samples (100 replicates)