(this note doesn’t include katie’s ppt and bhavin’s practical)

Monday (Austin Burt)

**Gene drive and its potential use for malaria control**

**Mendelian transmission**: equal, unbiased inheritance, no change in gene frequency

**gene drive**: preferential inheritance, will lead to spread in population, a ‘fifth force’ in population genetics, in addition to mutation, migration, drift, selection

**3 ways to drive:**

* killing the competition:
  + **t-haplotype in mice**: 4 inversions preventing recombination; homozygous sterile and/or lethal; lead to competition sperm premature acrosome reaction and/or flagellar dysfunction
  + **segregation distorter in drosophila**: failures in chromatin condensation
* over-replication:
  + homing endonuclease genes
  + transposable elements -- ‘jumping genes’: e.g. retroelements encode their own DNA polymerase (dna -> rna -> dna), DNA transposons (e.g. rapid spread of P elements in Drosophila)
* directed movement towards the germline:

gonotaxis: genes are more likely to enter germ instead of soma (or enter megaspore instead of polar bodies)

**potential use of gene drive in malaria control**

rationale:

* malaria continues to impose a huge burden: Africa infants and children die
* current interventions (nets, spraying, drugs) have saved millions but not enough to eliminate, drug-resistance and insecticide-resistance, cost more than funding available
* millions more set to die over coming years

biological background

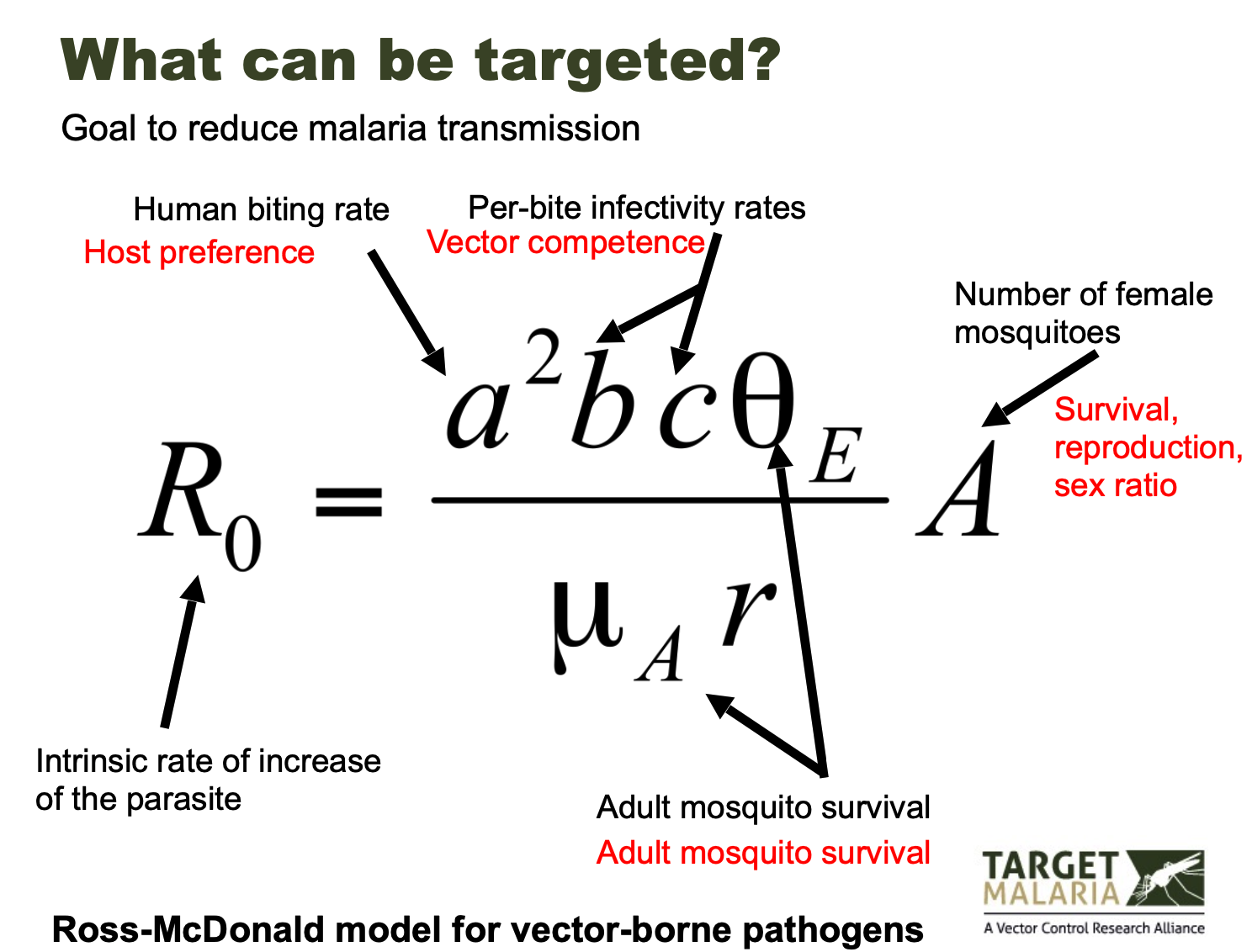
* malaria in Africa is largely rural
* there are 4 species of Plasmodium causing malaria in Africa
* most transmission is by 3 closely related Anopheles species
* only female mosquitoes bite people and transmit disease

What is gene drive?

* a natural process which we are learning to mimic
* can lead to the spread of genes that cause harm to the individuals carrying them
* makes for a potentially attractive new platform to control pests & vector-borne diseases

gene drive: two basic strategies

* population suppression: releasing modified mosquitoes into the population can cause transient or permanent population suppression
* population replacement: releasing modified mosquitoes into the population can lead to the spread of a gene that blocks malaria transmission
* or combine these 2 approaches together



How do we ‘make’ modified mosquitoes?

* plasmid helper RNA microinjected into eggs
* injected adults are crossed into wild type (hybrid)
* larvae are screened for green fluorescence to identify transgenics
* adult modified mosquitoes

Requirements for success

* genetic construct(s) with the required characteristics
* regulatory authorisation to release them
* public support / political will to release them
* sufficient local capacity to release them

Technical progress

* what works best in the computer

two gene drive strategies identified and confirmed as potentially useful:

* driving Y chromosome
* gene knock-out by homing: put nuclease gene in middle of target gene, lead to population-wide knock-out of target gene, e.g. female fertility gene knock-out can lead to population reduction.

Requires enzymes that recognise and cut specific DNA sequences. Over years have tried many types of enzymes (meganucleases, ZFNs, TALENs. Now using CRISPR, as much easier to use.

* can we make it in the lab
  + does the homing reaction occur in mosquitoes: experiments using a nuclease and its recognition sequence from yeast
  + can we design nucleases to home into endogenous mosquito genes: identified 3 female fertility genes (Ovary-specific chitin binding, Yellow G, Nudel) and designed CRISPR constructs against them, average homing rate across 3 genes is 94%
  + will they spread in a cage:

at the beginning yes, but show resistance later.

molecular & genetic analyses showed:

Resistance due to changes at target site that prevented cleavage and restored gene function.

All detected resistant alleles were in-frame insertions / deletions (no SNPs) from end-joining repair.

Changes due to nuclease activity, not pre-existing

* + can we avoid resistance:
    - target sequences less able to tolerate changes while maintaining function. (highly conserved across species)
    - the doublesex gene, involved in insect sex determination
    - homozygous knock-out females are sterile (and cannot blood feed)
    - no obvious effect on males
    - CRISPR construct gives high homing rates
    - led to population crash in cages without sign of resistance
  + scaling up cage size
    - Can mimic tropical light, temperature and humidity dynamics
    - Allows lower, more realistic densities and mating environment (swarming)
* does it work in the field

Tuesday (Hui, Tin-yu)

Evolutionary Modelling

objectives:

* build stochastic forward simulators in R
  + for the WF model
  + for a gene drive model
* use them to solve problems
  + i.e. monte-carlo methods

**Drift simulator**

* The WF model describes the change in allele frequency in a population due to genetic drift.
* During reproduction, an offspring is formed by sampling (with replacement) from the parent generation.

– famously modelled by the urn model

– binomial sampling of alleles

* Other assumptions include

– constant population size

– discrete generation

– random mating

– closed population

– no mutation, no selection

* one locus, two alleles (labelled as 0 and 1)
* Diploid individuals
* parameters:

– 𝑁: the (constant) population size

– 𝑝0: the initial frequency of allele “0”

– 𝑡: number of generations

* Initialisation

– what are the object and data types (lists/matrices/vectors, numeric/character/logical)?

– 𝑁 individuals, each carrying two alleles, matrices maybe? – how do we utilise the input arguments?

– the initial condition(s) of the simulation?

* Propagation

– use information from gen i to sample alleles at gen (i+1)

– or from gen (i-1) to i

– use a loop for recursive calculation?

– constant 𝑁 throughout

– for()? sample()?

* Output

– what are the outputs?

– return() and exit the function

# GENETIC DRIFT SIMULATION

sim\_genetic\_drift<-function(p0=0.5, t=10, N=50)

{

# INITIALISATION

population <- list()

length(population) <- t+1

# give names to the elements of population

names(population) <- rep(NA, t+1)

for (i in 1:length(population)){

names(population)[i] <- paste(c('generation', i-1), collapse = '')

}

#also keep track on the allele freq over time, as a vector

allele.freq <- rep(NA, t+1)

k <- ceiling(2\*N\*p0)

population[[1]] <- matrix(sample(c(rep(0,k), rep(1, 2\*N-k))), nr=2)

#the initial allele freq

allele.freq[1] <- sum(population[[1]]==0)/(2\*N)

# PROPAGATION

for (i in 1:t)

{

#sample alleles for the next gen

#based on the allele.freq at the current gen

population[[i+1]]<- matrix(sample(0:1, size=2\*N,

prob = c(allele.freq[i], 1 - allele.freq[i]), replace=TRUE),

nr=2)

#the allele freq at the next gen

allele.freq[i+1] <- sum(population[[i+1]]==0)/(2\*N)

}

#OUTPUT

return(list(population=population, allele.freq=allele.freq))

}

* Some probabilistic questions
  + The WF model suggests genetic drift will not change the mean allele frequency but will increase the uncertainty (variance) surrounding it
  + what is the mean and variance of allele frequency after t=10 generations, given p0=0.5, N=200?
  + Method #1 not good

– Mathematical derivations

– Work out the distribution of the allele frequency, calculate the mean and variance of the associated random variables

– Diffusion approximation, Markov matrices and processes etc.

* + Method #2 good

– Monte-Carlo methods

– The average results from repeated simulation

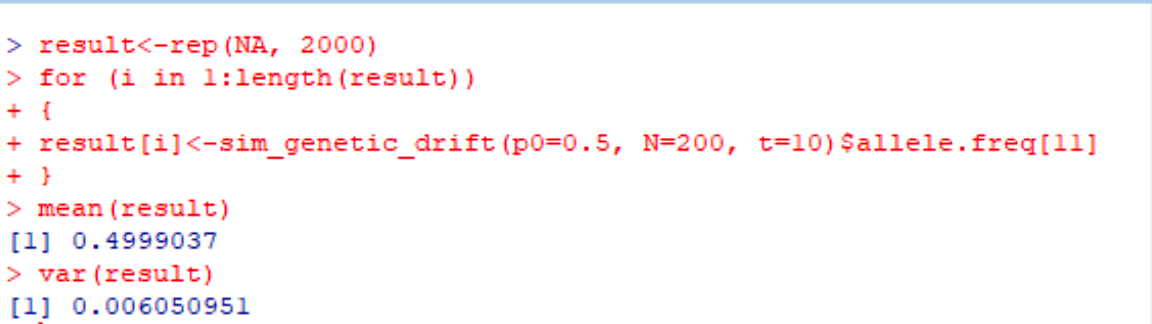
* Monte Carlo simulations
  + Suppose our aim is to approximate the mean and variance of allele frequency due to genetic drift

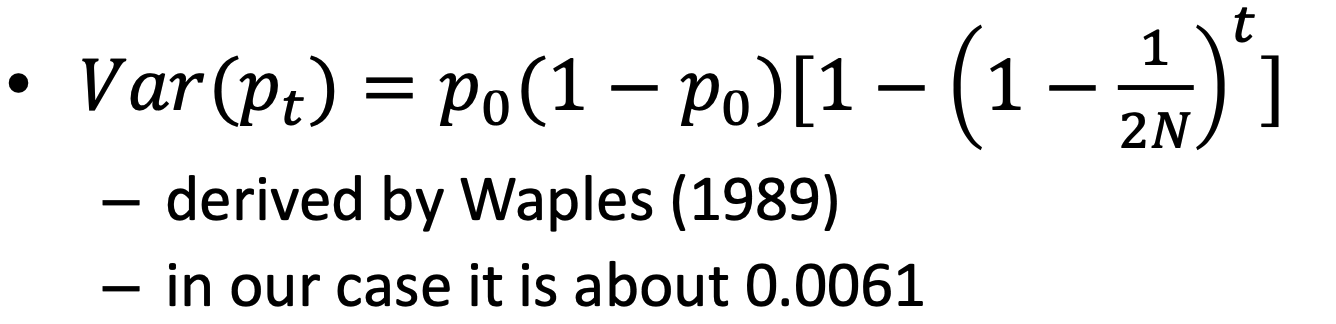
– simulate the WF model with known parameters 𝑝0 = 0.5, 𝑁 = 200, and 𝑡 = 10

– record the allele frequency at the final generation

– repeat the simulation, independently, for 10,000 times (say) to obtain 10,000 such final allele frequencies

– calculate the sample mean and variance of these 10,000 final frequencies, and these are your empirical answers





* The result from MC simulation will converge to the “true answer” if the number of independent simulations → ∞

– the stochastic nature means a different answer is expected if you rerun the MC simulation

– intrinsic variance ∝ 1/(𝑛𝑢𝑚𝑏𝑒𝑟 𝑜𝑓 𝑠𝑖𝑚𝑢𝑙𝑎𝑡𝑖𝑜𝑛𝑠)

**Gene drive simulator**

* similar to the WF model but with ‘add-ons’

– drive

– selection

– fluctuating population size

* two types of alleles

– 0 is the wildtype allele (WT)

– 1 is the artificially introduced transgene (TG)

* Drive: super-Mendelian inheritance of a gene (the transgene in our case), resulting in unbalanced number of gametes being produced by the heterozygotes

– an individual carrying 01 heterozygote produces gamete 0 with proportion (1 − 𝑑), and gamete 1 with proportion 𝑑, 𝑑 > 50% is required for TG to drive

– this allows TG to spread even with a low initial frequency

* Selection:

– 11 homozygote does not survive till adulthood – produces no offspring

* Combining both effects

– 00 homozygote adults produce WT gamete with probability 1

– 01 heterozygote adults produce WT gamete with probability (1 − 𝑑), and TG gamete with probability 𝑑

– 11 homozygotes produce no gametes

* Let {𝑥00 , 𝑥01 , 𝑥11 } be the number of 00 01 11 individuals carrying 00, 01, 11 genotypes at the parental generation
* What are the gametic frequencies for WT and TG?

– proportion of WT gametes = [𝑥00∗1+𝑥01∗(1−𝑑)] / [𝑥00+𝑥01]

– proportion of TG gametes = 𝑥01∗𝑑 / [𝑥00 +𝑥01]

* Population dynamics: the population size is no longer constant because of the two forces acting on the it:

– the lethal 11 homozygote, suppressing the population

– the lack of intra-species competition when population size is small, reverting it back to the carrying capacity

* For the second point we introduce the Beverton-Holt (1957) model to regulate the population size:

– Nt+1 = R0\*Nt / [1+Nt/M] with two extra parameters

– discrete version of logistic growth, (𝑅0 − 1) \* 𝑀 is the carrying capacity.

– “density dependence”

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

# GENE DRIVE SIMULATOR. WRITTEN IN R

# CMEE MSc 2021 TIN-YU HUI

# DRIFT: WRIGHT-FISHER MODEL

# SELECTION: 11 HOMOZYGOTES ARE LETHAL

# DRIVE: SUPER-MENDELIAN INHERITANCE OF TG WITH RATIO d, d>0.5

# 0=WT ALLELE; 1=TG ALLELE

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

# INPUT PARAMETERS:

# q0: INITIAL RELEASING FREQ OF TG

# N0: INITIAL POPULATION SIZE (DIPLOID)

# d: TRANSMISSION RATE OF TG (d>0.5 IN ORDER TO DRIVE)

# t: NUMBER OF GENERATIONS TO SIMUATE FORWARD IN TIME

# R0, M: TWO EXTRA PARAMETERS FOR THE BEVERTON-HOLT MODEL FOR POPULATION REGULATION

sim\_gene\_drive<-function(q0=0.05, d=0.6, t=10, N0=500, R0=2, M=500)

{

# SOME CHECKS ON THE INPUT PARAMETERS (OPTIONAL)

if (q0<=0 || q0>0.5)

{stop('PLEASE MAKE SURE THAT 0<q0<0.5!')}

if (d<=0.5 || d>=1)

{stop('PLEASE MAKE SURE THAT 0.5<d<1!')}

# INNER FUNCTIONS. THESE INNER FUNCTIONS ARE ONLY VISIBLE WITHIN sim\_gene\_drive()

# 1) THE BEVERTON-HOLT MODEL. ceiling() TO ROUND UP. RETURN NEW POPULATION SIZE.

bh<-function(N, R0, M)

{return(ceiling(R0\*N/(1+N/M)))}

# 2) RETURN THE COUNTS FOR 00, 01, 11 GENOTYPES

count\_genotype<-function(x)

{

temp<-apply(x, 2, sum)

return(c(sum(temp==0), sum(temp==1), sum(temp==2)))

}

# INITIALISE

# CREATE A LIST TO STORE ALL THE ALLEIC CONFIGURATIONS

population<-list()

length(population)<-(t+1)

for (i in 1:(t+1))

{names(population)[i]<-paste(c('generation', i-1), collapse='')}

# ALSO CREATE TWO VECTORS TO STORE THE POPULATION SIZES AND THE FREQ OF TG OVER TIME

population.size<-rep(NA, t+1)

TG.freq<-rep(NA, t+1)

# INITIAL POPULATION SIZE AND TG FREQ

population.size[1]<-N0

TG.freq[1]<-q0

# WE WILL RELEASE k TRANSGENIC MOSQUITOES, WHO CARRY 01 HETEROZYGOTE

# WHICH MEANS AT GEN 0 THERE ARE (N0-k) WT MOSQUITOES WITH 00 HOMOZYGOTES

k<-ceiling(2\*N0\*q0)

population[[1]]<-cbind(matrix(c(0,0), nr=2, nc=N0-k), matrix(c(0,1), nr=2, nc=k))

# CALCULATE THE GENOTYPE COUNTS (WE'LL REUSE THE VECTOR genotype IN THE FOR LOOP)

genotype<-count\_genotype(population[[1]])

# PROPAGATION

for (i in 1:t)

{

# CALCULATE THE NEW POPULATION SIZE. ONLY genotype[1]+genotype[2] WILL SURVIVE TILL ADULTHOOD

population.size[i+1]<-bh(genotype[1]+genotype[2], R0, M)

# EARLY EXIT CONDITION 1, IF POPULATION SIZE DROP TO 1

if (population.size[i+1]<=1)

{

print(paste(c('Oops! The population crashed after generation ', i-1), collapse=''))

return(list(population=population[1:i], population.size=population.size[1:i],

TG.freq=TG.freq[1:i]))

}

# EARLY EXIT CONDITION 2, IF THERE IS NO MORE TG ALLELE

if (genotype[2]+genotype[3]==0)

{

print(paste(c('Oops! TG allele went extinct at generation ', i-1), collapse=''))

return(list(population=population[1:i], population.size=population.size[1:i],

TG.freq=TG.freq[1:i]))

}

# CALCULATE TG GAMETIC FREQ

TG.gametic.freq<-d\*genotype[2]/(genotype[1]+genotype[2])

# SAMPLE THE NEXT GENERATION

population[[i+1]]<-matrix(sample(0:1, size=2\*population.size[i+1],

prob=c(1-TG.gametic.freq, TG.gametic.freq), replace=T), nr=2)

# CALCULATE NEW GENOTYPE COUNTS AND TG FREQ

genotype<-count\_genotype(population[[i+1]])

TG.freq[i+1]<-(0.5\*genotype[2]+genotype[3])/population.size[i+1]

}

# OUTPUTS. RETURN A BIG LIST OF EVERYTHING

return(list(population=population, population.size=population.size, TG.freq=TG.freq))

}

# TEST RUN (THIS LINE CAN BE DELETED)

sim\_gene\_drive(q0=0.05, d=0.6, t=10)

Practical

* releasing strategy (q0): guarantee TG survive > 95%
* construct design (d): depress population size quickly
* targeting different population profiles [same N0 different R0 (intrinsic growth rate) and M, same carrying capacity (R0-1)\*M]: R0 smaller, TG depress population size better
* equilibrium freq (equilibrium TG frequency does not depend on the initial frequency):
  + One property of this gene drive is that the equilibrium TG frequency depends on 𝑑 only

– unrelated to initial frequency (as long as it survives)

– unrelated to population sizes etc.

* + The equilibrium frequency of TG ≈ (2𝑑 − 1) – there is also an equilibrium population size
  + To visualise this feature, we can simulate with different releasing frequencies, and track the TG frequencies over time. They will all converge to the same value.

Possible extensions include

* Migration with multiple populations
* Seasonal population dynamics
* Heterozygote (dis)-advantages
* The arise of resistant allele
* Drive systems with genetic linkage (multiple loci), multiple drives
* Computationally demanding

– repeated simulations

– memory issue when 𝑁 goes large

* Efficient coding

– vectorisation in R

– C/C++? C with R interface?

* Parallel implementation

– R packages for multicore cpu (doParallel etc) – College’s HPC

– GPU?

doParallel package

* A built-in R package to replace the generic for() by the parallelised foreach() loop

– with some grammatical changes

– similar to mclapply()

* Iterations are sent to different cpu cores and executed simultaneously (but also independently)

– ideal for MC simulations

* Parallel random number generation is “handled” by the package

– unlike HPC, where it is your responsibility to make sure different nodes are using different random seeds/states

* Smaller tasks are less likely to be benefited from parallelisation because of overhead

– performance gain is also problem-specific

# PARALLELISED VERSION doParallel package

# LOAD THE PACKAGE

require(doParallel)

# CREATE A LOCAL “CLUSTER” OBEJECT cl. THE NUMBER INSIDE THE BRACKET

MUST NOT EXCEED YOUR CPU CORE COUNTS. REGISTER THE CLUSTER BEFORE USE.

cl<-makeCluster(6)

registerDoParallel(cl)

# foreach

# .combine IS AN ARGUMENT SPECIFYING HOW RESULTS FROM DIFFERENT RUNS

ARE COMBINED. POSSIBLE WAYS ARE c, cbind, rbind. DEFAULT IS A LIST.

# REMEMBER %dopar%

result<-foreach(i=1:1000, .combine='c') %dopar% {

temp<-rchisq(30, df=1)

return(sum(temp))

}

# STOP THE CLUSTER AFTER USE

stopCluster(cl)

Thursday (Bhavin)

what is evolution?

* Life: Reproducing organism carrying information, e.g. in DNA, about how to survive in an “environment”
* Selection: those organisms that have the best information about surviving/reproduce best/fastest (survival of the fittest)
* Mutation: random changes in information
* Mutation + Selection --> adaptation to best information
* Random Genetic Drift: Stochasticity due to finite size of populations -> sometimes populations adapt to sub-optimal information

Basic evolutionary force: selection

* fitness w, mutant with growth advantage s (selection coefficient, key parameter)
* wild type: growth rate w
* mutant type: growth rate: w(1+s)

Basic evolutionary force: mutation

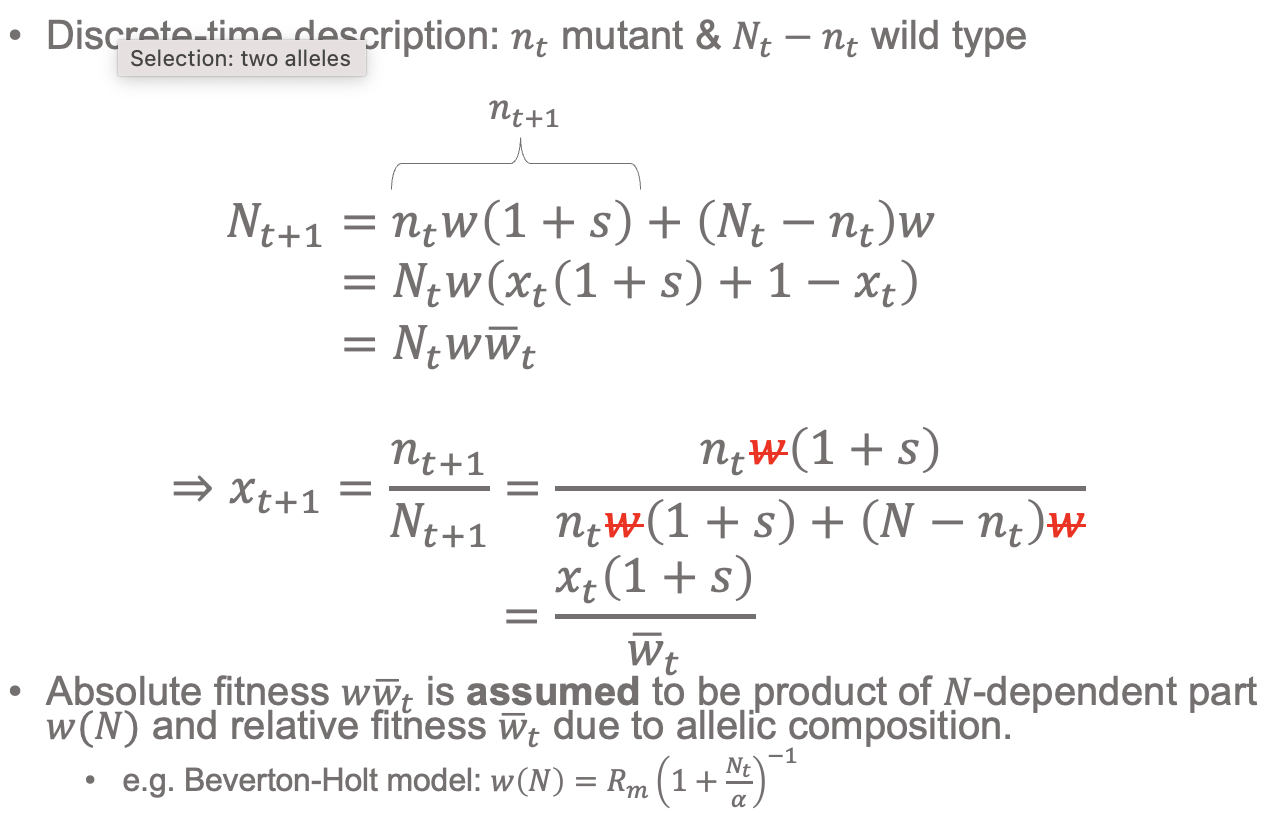
* wild type → mutant: μ
* mutant → wild type: v

Basic evolutionary force: drift

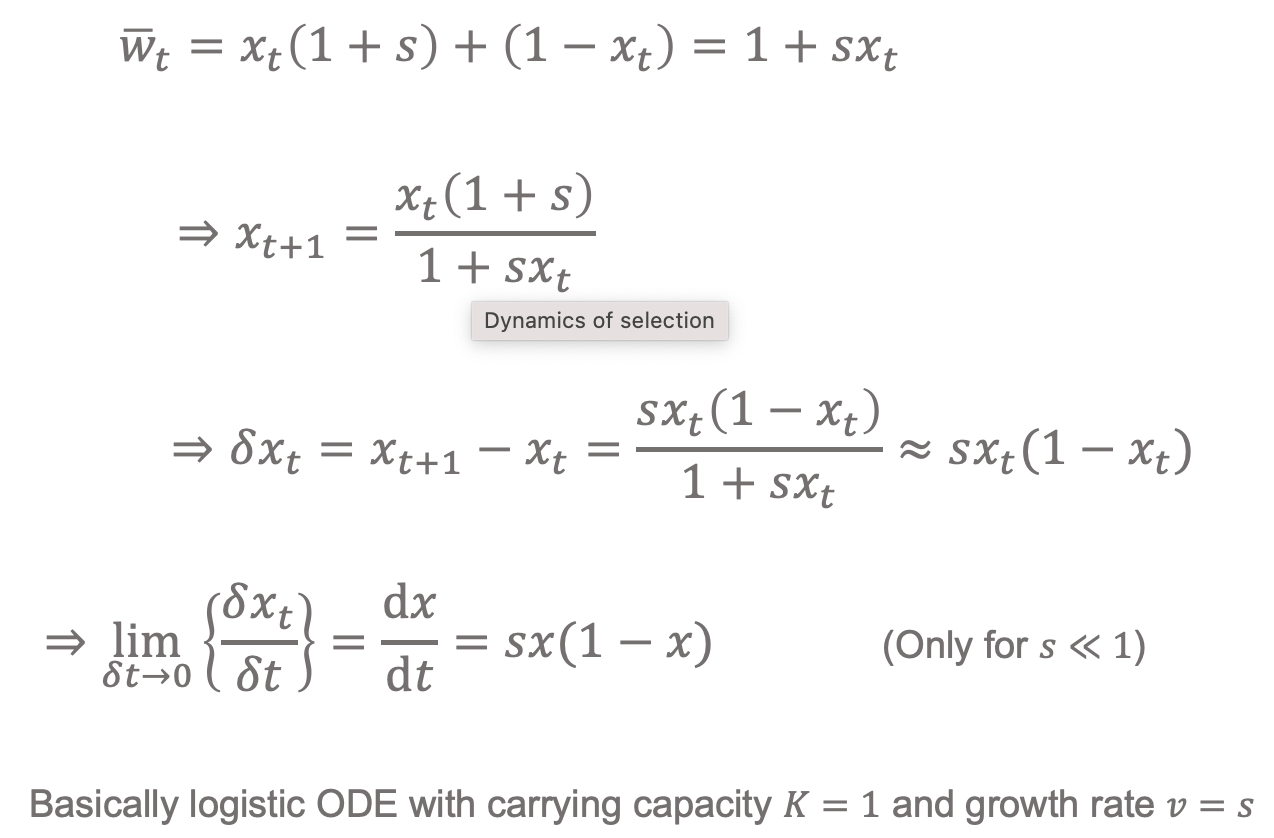
* key parameter: effective population size, N

Recombination & linkage - focus on single locus unlinked sites

Selection: two alleles



S << 1 & ODE description



Dynamics of selection

* s-shape curve
* dx/dt = sx(1-x)
* x(t) = (x0 est) / (x0 est + 1 - x0 )
* t\* = (1/s) ln[(1/x0) - 1] (mutant gene frequency reached 0.5)
* x0 smaller, t\* bigger
* s smaller, t\* bigger

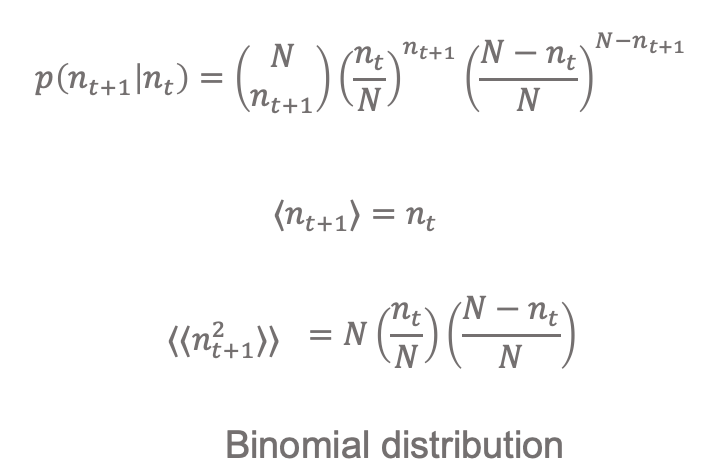
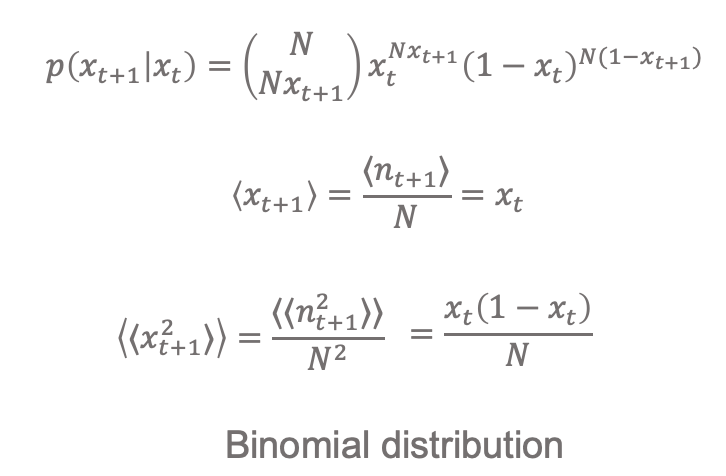
Dynamics of mutant (deterministic)

* wild type → mutant: μ
* mutant → wild type: v
* dx/dt = - vx + μ(1-x)
* x\* = μ/(μ+v) (equilibrium mutant frequency)
* x = x\* + (x0 - x\*)e^-(μ + v)t

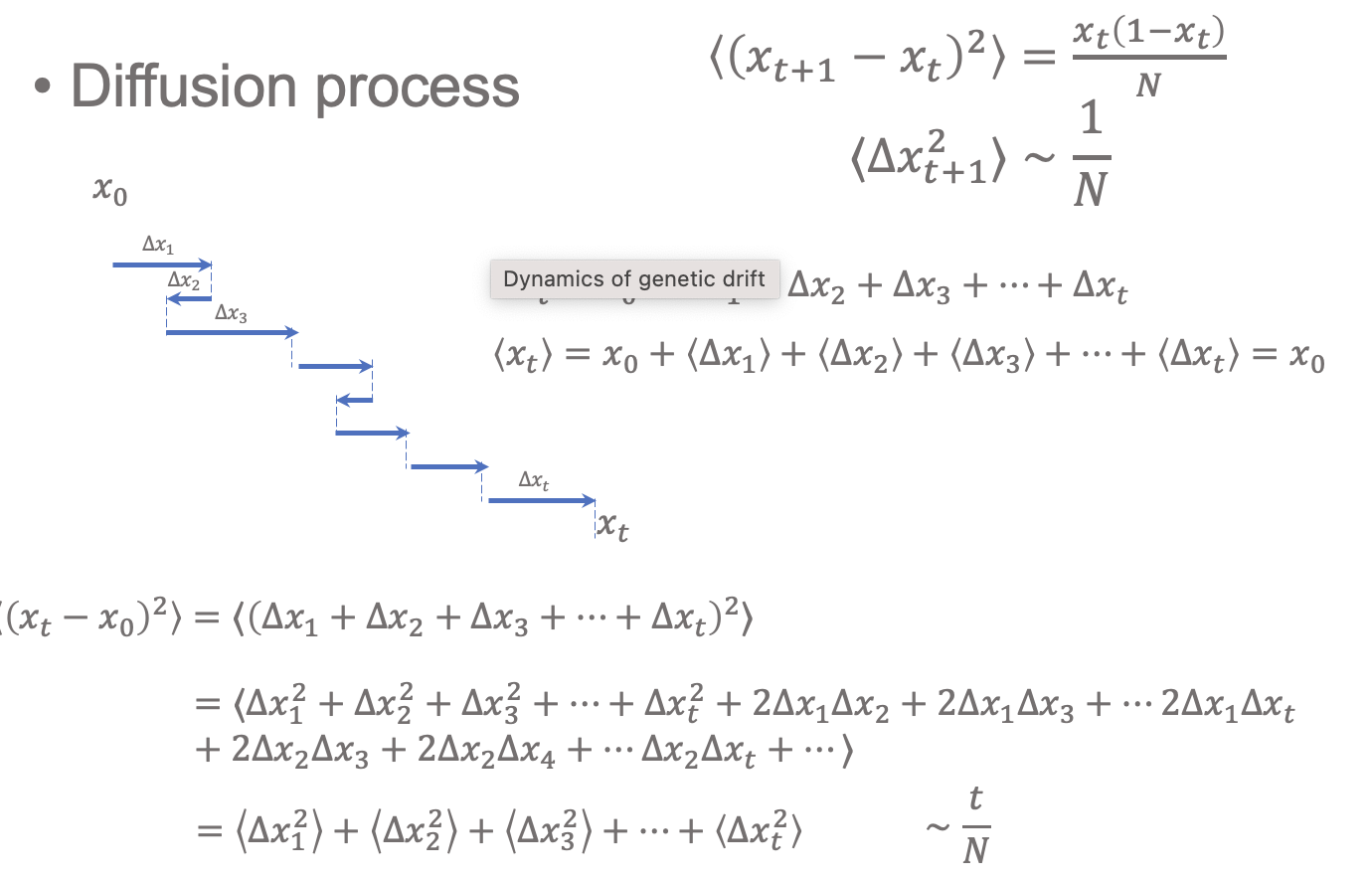
Deterministic to Stochastic evolutionary dynamics

* key concept exact predictability vs probability distribution
* bigger N, smaller drift effect

Random Genetic Drift (Wright-Fisher model)

* no selection, no mutation
* 
* <nt+1> expectation of nt+1
* <<n2t+1>> square error
* gene frequency version
* 
* bigger N, smaller square error, change in frequency between generations becomes smaller

Dynamics of genetic drift



Time to fixation or extinction t\*

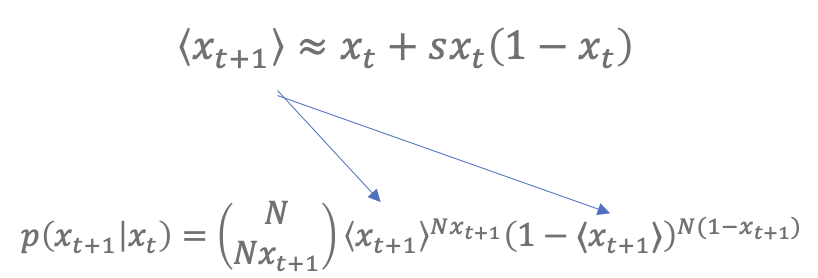
* <xt+1> = xt
* <(xt\*-x0)2> ~ t\*/N ~1
* ⇒ t\* ~ N

Summary of properties of drift

* genetic drift stronger in smaller population
* diffusion process where <(xt\*-x0)2> ~ t/N
* eventually drift will leave only a single variant (fixation or loss): this takes time t\* ~ N

Drift + Selection (Wright-Fisher model)

<xt+1> ~ xt + sxt(1-xt) (s << 1)



Drift + Selection + Mutation (Wright-Fisher model)

<xt+1> ~ xt + sxt(1-xt) + μ(1-2xt) (μ,s << 1)

Nμ key parameter for dynamics

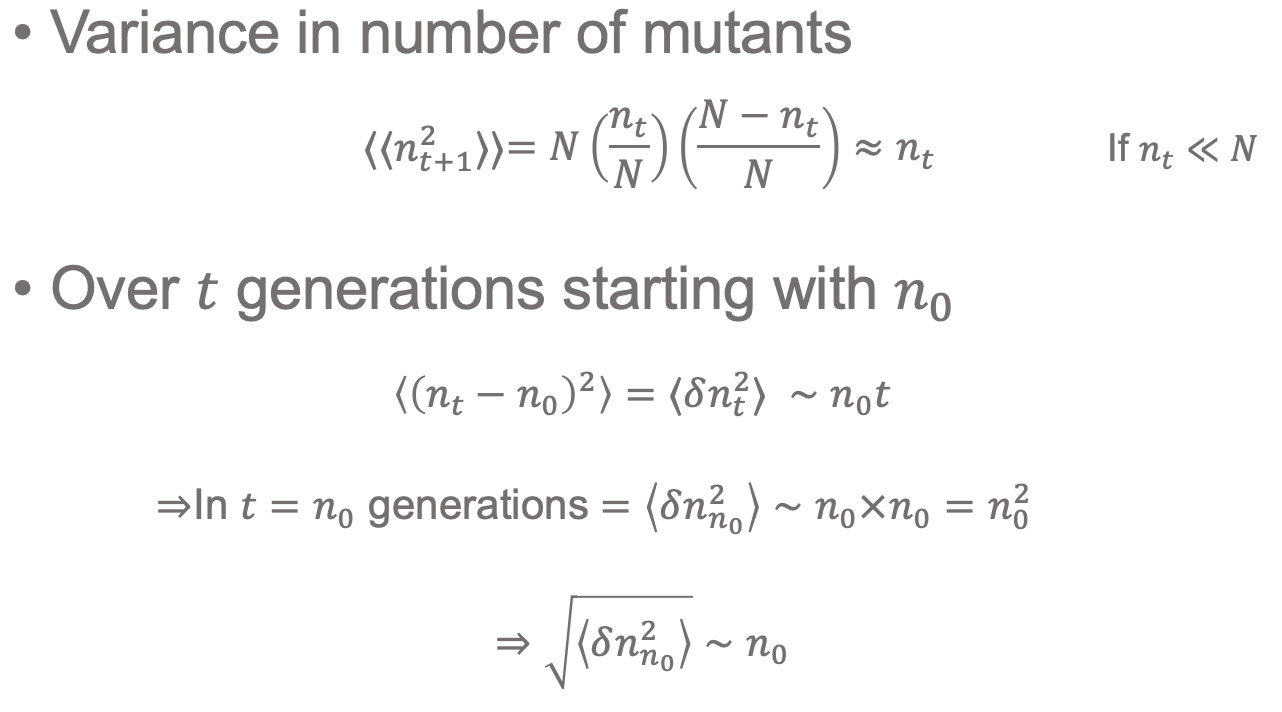
* Neutral mutation - Nμ steadily decreased

Interplay between selection and drift

* Even in large population sizes drift can be important
* All mutants start at frequency 1/𝑁
* ploss=(1−1/𝑁)N≈(𝑒−1/𝑁 )N = 𝑒^(−1) ≈ 0.37
* Selection does not change picture: 𝑝\_𝑙𝑜𝑠𝑠≈𝑒−(1+𝑠) ≈𝑒-1
* New mutants have to survive loss

Need to grow to a critical frequency → thereafter growth ~deterministic

Stochastic growth dynamics



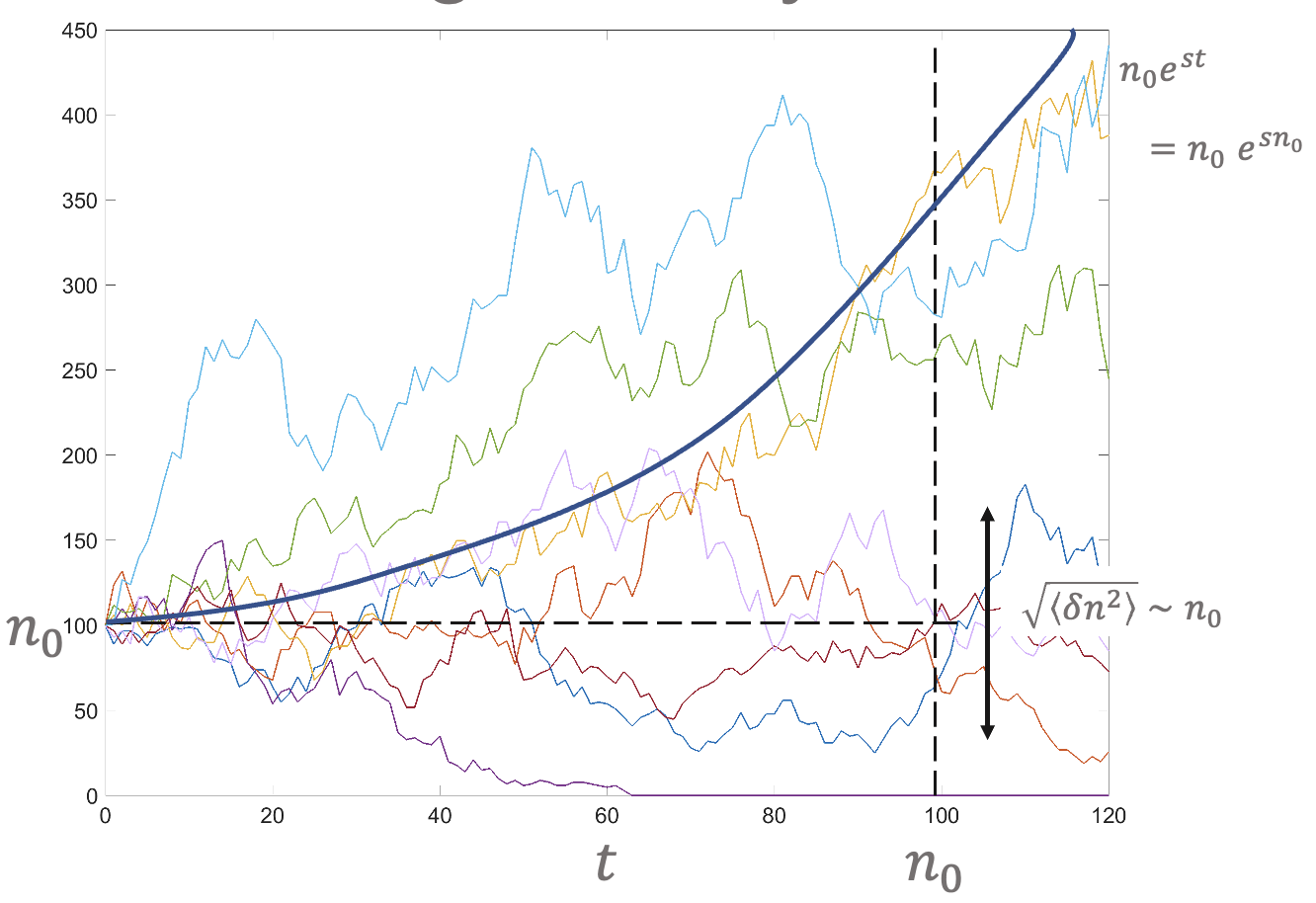
* Mutant starts with 𝑛 individuals
* Drift will take of order 𝑛 generations to change number of mutants of order 𝑛⇒ change in frequency |𝛿𝑥𝑁 |≈𝑛/2𝑁

|𝛿𝑥𝑁 / 𝛿t| ≈ 1/2𝑁

* Per generation selection will give a change in freq (d𝑥𝑠)/𝑑𝑡=𝑠𝑥(1−𝑥), which for 𝑥≪1

𝛿𝑥s/𝛿t ≈ sx

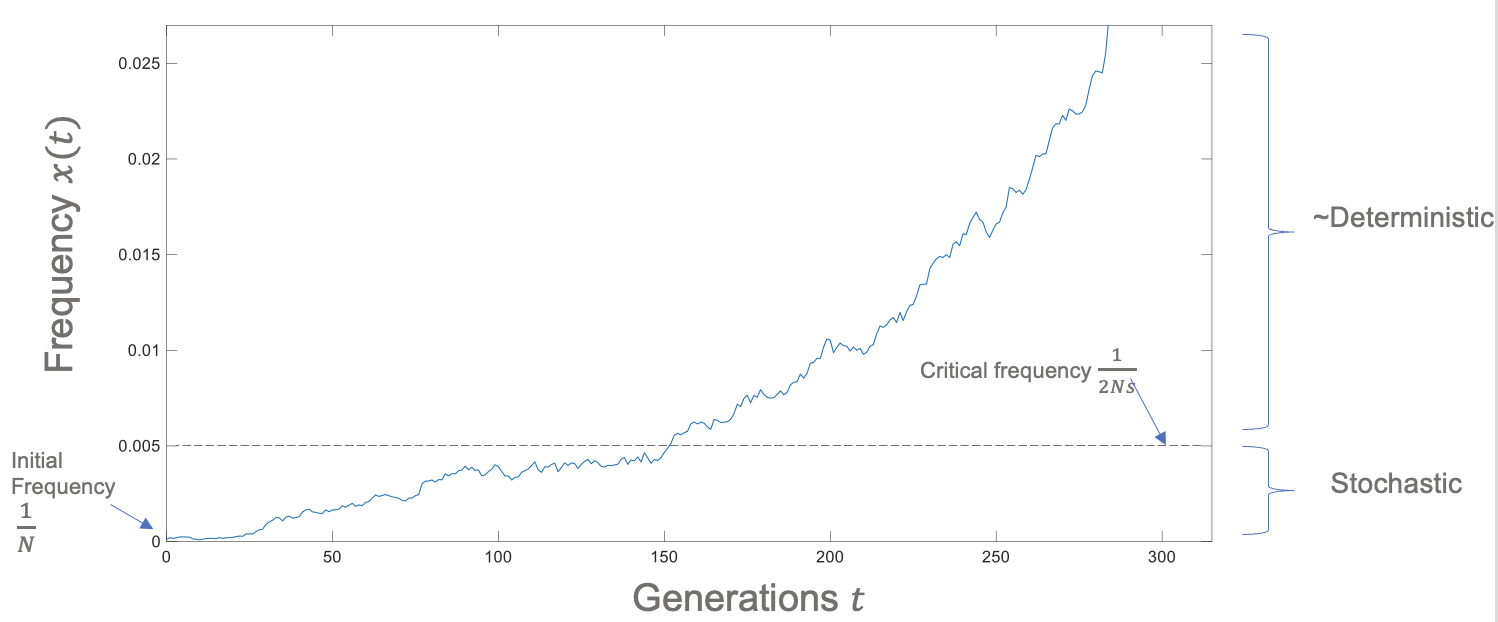
* Selection dominates drift when |(d𝑥𝑠)/d𝑡| ≫|(d𝑥𝑁)/d𝑡|
* Establishment frequency 𝑥\*∼1/2𝑁𝑠



\* blue line is selection, arrow is drift effect scope, n0/2 is |𝛿𝑥𝑁 | with n0 individuals starting

Establishment

* Mutant with selective advantage s (2Ns >> 1)



* Time for mutant to rise by drift to frequency 𝑥\*=1/2𝑁𝑠
* This corresponds to a mutant population of size

n\* = Nx\* = 1/2s

* which takes time

τest∼n\*=1/2s

Probability of fixation of neutral mutant (Nμ << 1)

* Given an initial frequency 𝑥\_0=1/𝑁 what is ultimate probability of fixation
* Neutral case: Everyone identical except labelling ⇒𝑝𝑓𝑖𝑥=1/𝑁
* Selection s: if 2𝑁𝑠≫1, then once mutant has established, it will fix with probability close to 1

⇒𝑝𝑓𝑖𝑥≈𝑝𝑒𝑠𝑡

For 𝑥 < 1/(2𝑁𝑠) allele freq changes dominated by drift

𝑝𝑒𝑠𝑡≈Pr⁡(𝑚𝑢𝑡𝑎𝑛𝑡 𝑟𝑒𝑎𝑐ℎ𝑒𝑠 𝑠𝑖𝑧𝑒 𝑛\*=𝑁𝑥\*=1/2𝑠)

⇒mutant is just one out of neutral sub-population of size n\*

⇒pfix≈pest≈1/n\* =(1/2s)-1 = 2s

* What if x\* = 1/(2Ns) >= 1?
  + selection is weak and mutant is ~ neutral ∀ 𝑥: 0≤𝑥≤1

⇒ if 2𝑁𝑠≤1 ⇒𝑝𝑓𝑖𝑥≈1/𝑁

⇒ mutants with 𝑠≤1/2𝑁 are selectively neutral

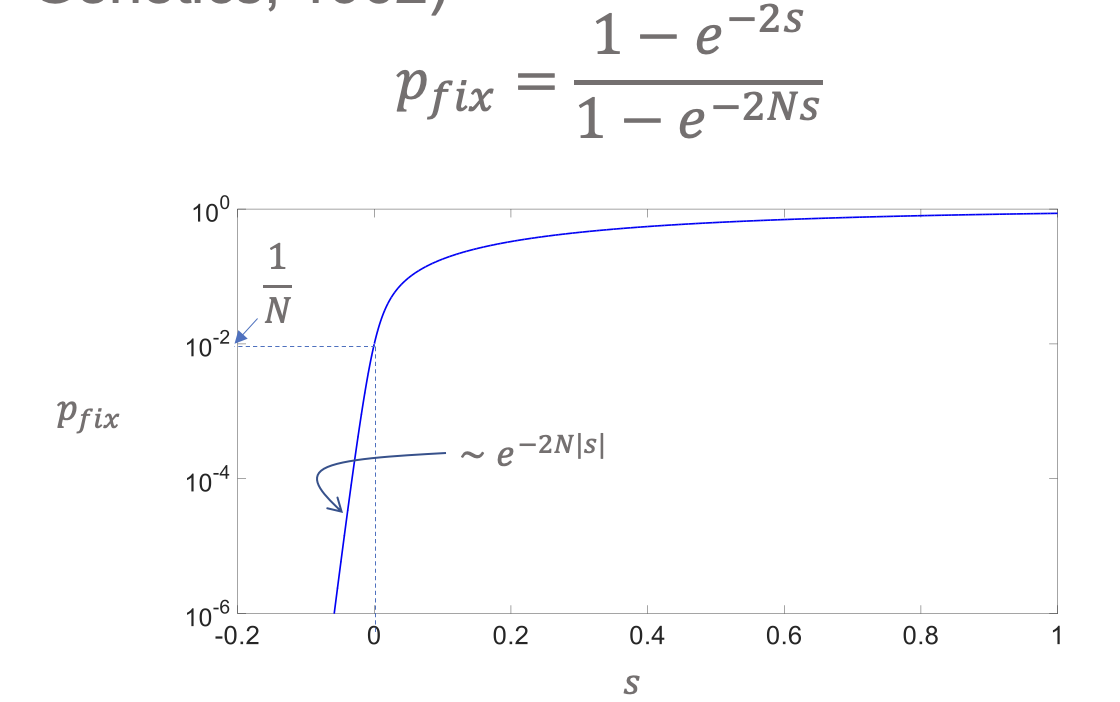
* + 1/2𝑁 is a natural scale for fitness differences

Summary of heuristic analysis of establishment

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Critical frequency | Critical size | Probability of establishment | Time to establishment | Rate of establishment |
| x\*=1/2Ns | n\*=1/2s | pest≈2s≈pfix | τest≈1/2s | kest=1/τest ≈2s |

Probability of fixation (Nμ << 1)

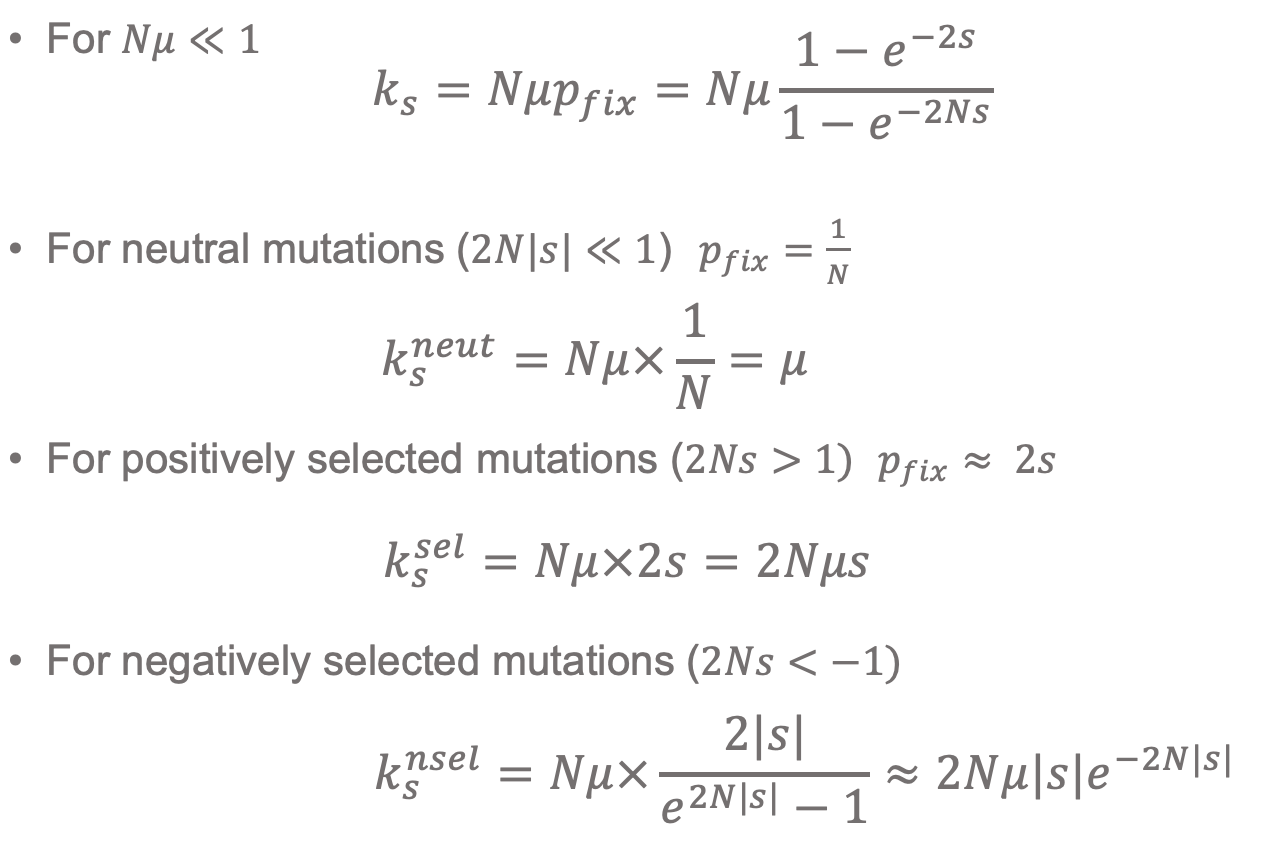
* Kimura formula for general s, N(Kimura, Genetics, 1962)



Substitution vs mutation rate

* Mutations whether beneficial, neutral or deleterious, arise at the same rate (individual level)
* Substitution rate does depend on whether allele is beneficial, neutral or deleterious (population level)

Substitution rate



Infinite sites model

* If Nμ≪1, on short time scales mutations at the same site twice are unlikely as Nμ≫(Nμ)2
* For any long stretch of DNA, each new mutation gives rise to a new allele/sequence

Tests for selection: “dN/dS”

* If the substitution rate depends on 𝑠 maybe we can determine if selection is occurring in a particular gene making infinite alleles assumption
* If we knew the mutation rate 𝜇 then

if 𝑘s<𝜇 → negative selection and 𝑘s>𝜇 → positive selection

* Need an empirical “yardstick”!
* In coding genes, expect synonymous codon changes to be neutral (not always true!) ⇒𝑘𝑠syn=𝜇
* Measure substitution rates of non-synonymous 𝑘𝑠nsyn mutations wrt synonymous rate 𝑘𝑠syn in same gene
* Typically 𝑘𝑠nsyn≡"𝑑𝑁“ and 𝑘𝑠syn≡"𝑑𝑆“

⇒"𝑑𝑁/𝑑𝑆“ test but 𝑑’s are not differentials!!

* Also know as 𝐾a/𝐾s

|  |  |  |
| --- | --- | --- |
| s<-1/2N | -1/2N<s<1/2N | s>1/2N |
| ksnsel≈2Nμ|s|e-2N|s| | ksneut=μ | kssel=2Nμs |
| ksnsyn/kssyn≈2N|s|e-2N|s| < 1 | ksnsyn/kssyn = 1 | ksnsyn/kssyn = 2Ns > 1 |

Issues with dN/dS

* Not all synonymous changes are neutral (e.g. some codons can lead to faster replication)
* If only a few sites in a gene are under selection can be difficult to detect
* Many many papers still use dN/dS on intrapopulation variation!!
  + This is not correct!
  + dN/dS should only be used to compare substitutions between distinct populations or a distant outgroup
  + A mutation is not a substitution
  + A polymorphism is not a substitution