5. Population genetics: mutations, selection, drift - solutions

AUTHOR
EPFL - SV - BIO-463

PUBLISHED
March 19, 2024

Exercise 1 - Determining mutation rates: the fluctuation test

1. Each division event creates a cell (as it gives two cells from one cell). The number of cells per culture is N0 before growth and Nt after growth. Thus the number of division events occurring in the growth process is Nt-N0. Assuming Nt>>N0, it is approximately equal to Nt. Since m is the expected (average) number of mutation events that occurred in each culture, we have p=m/(Nt-N0) and thus approximately p=m/Nt.

2. The values given by webSalvador satisfy p=m/Nt. Since p=m/Nt, if Nt is 10 times larger but the list of the number of mutants in each culture is the same, we expect to estimate a 10 times smaller value of p. This is what happens using in the webSalvador estimate.

3. With twice more cultures and the exact same list of numbers of mutants obtained twice, we expect to have smaller uncertainties on the estimated parameters. This is indeed what happens in the webSalvador estimate.

4. If a plating efficiency epsilon=0.1 is assumed instead of epsilon=1, this means that the mutation probability was higher - it gave rise to the same number of mutants but not all of them were observed. This is indeed what happens in the webSalvador estimate.

5. Assuming w=1.5 instead of w=1 means that mutants divide faster. Thus, to obtain the same number of mutants, we have to have a smaller mutation probability. This is indeed what happens in the webSalvador estimate.

Exercise 2 - Drift and selection: simulating the Wright-Fisher model

1.

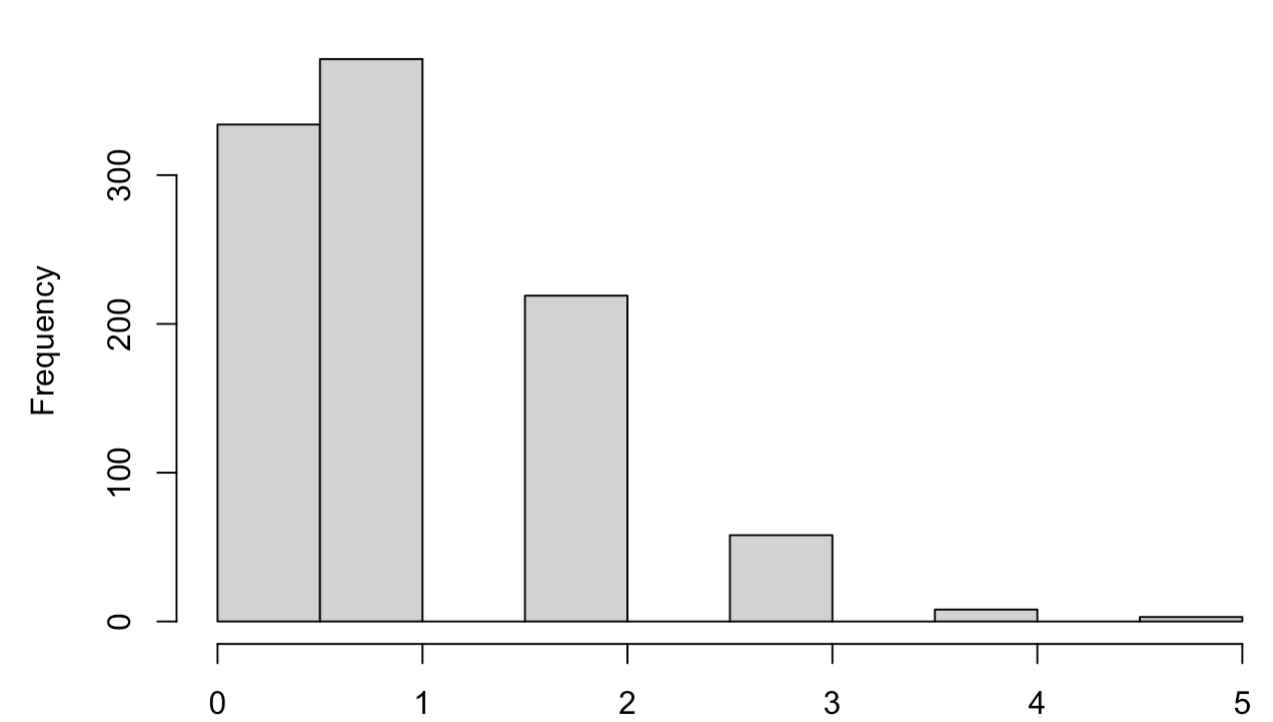
```
size=10
prob=0.1
n=1000
vals=rbinom(n, size, prob)
mean(vals)
```

```
[1] 1.037
```

```
sd(vals)
```

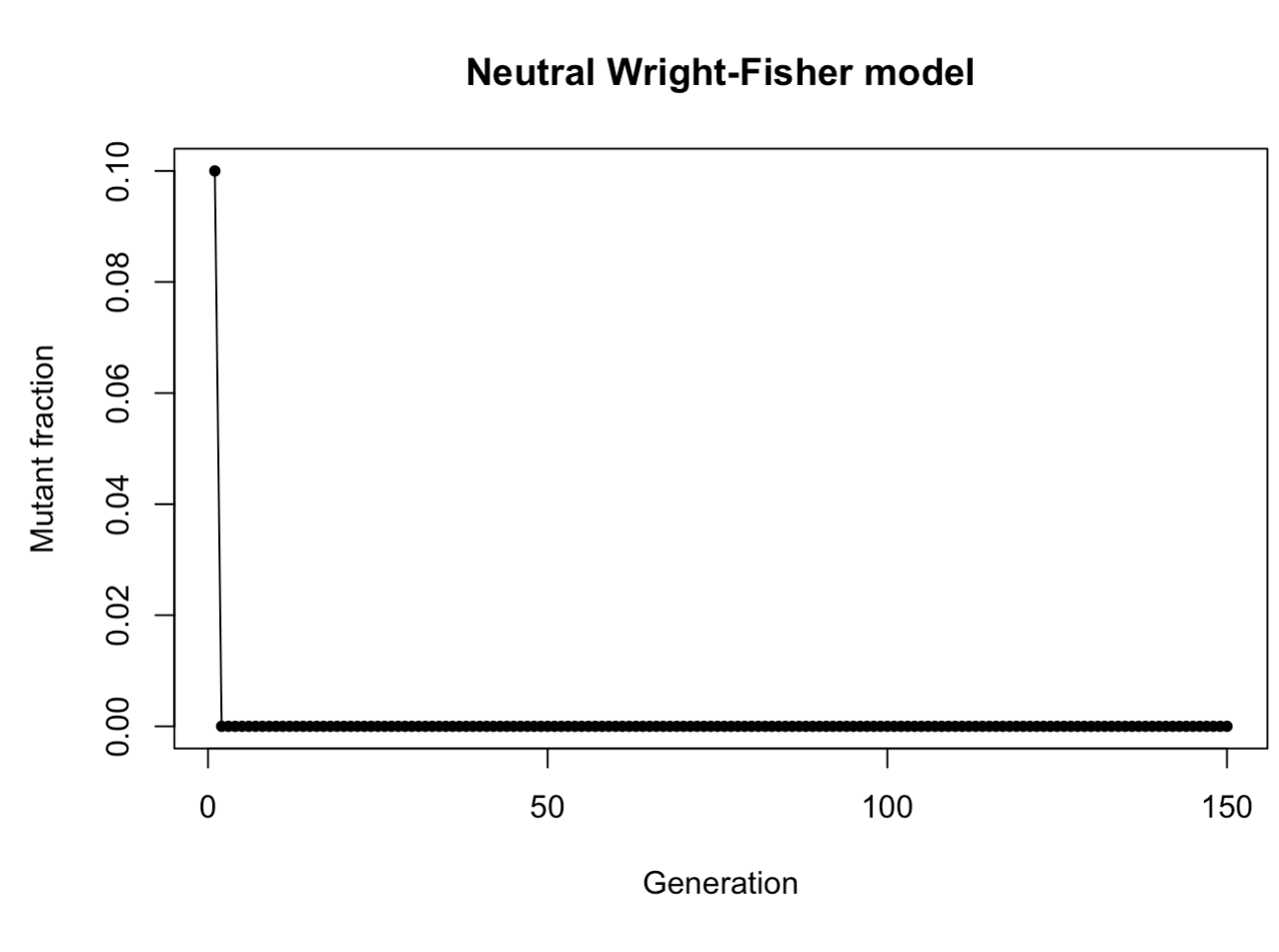
```
[1] 0.9510707
```

```
h1 = hist(vals, main='', xlab='Number of successes')
```



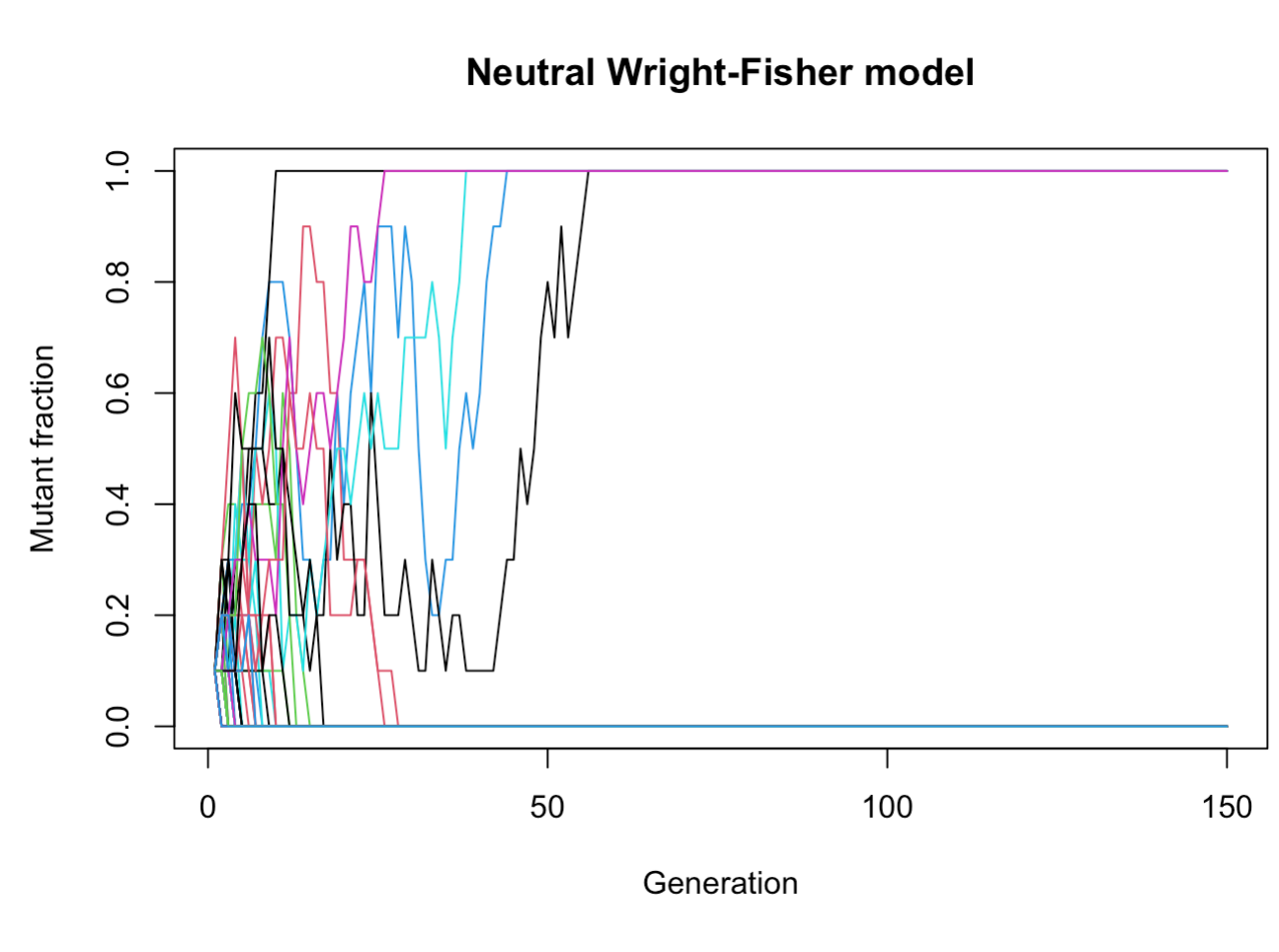
2. The trajectory we observe is a random walk. There can be large differences between different individual realizations. After a sufficient time, the mutant fraction hits 0 or 1 and then remains there - this corresponds to fixation of one type and extinction of the other. Most often, the ultimate state is extinction of the mutant, and it tends to happen quite fast. In fact, in the neutral case, the probability of fixation of the mutant strain is equal to the initial number of mutants divided by the population size, i.e. mutants_ini/size. It is equal to 1/10 here.

```
size=10
mutants_ini=1
n_gen=150
mutant_frac = numeric(n_gen)
mutant_frac[1]=mutants_ini/size
for(j in 2:n_gen){
  mutant_frac[j]=rbinom(1, size, mutant_frac[j-1])/size
}
plot(1:n_gen, mutant_frac, pch=20, type="o", main='Neutral Wright-Fisher model'
```



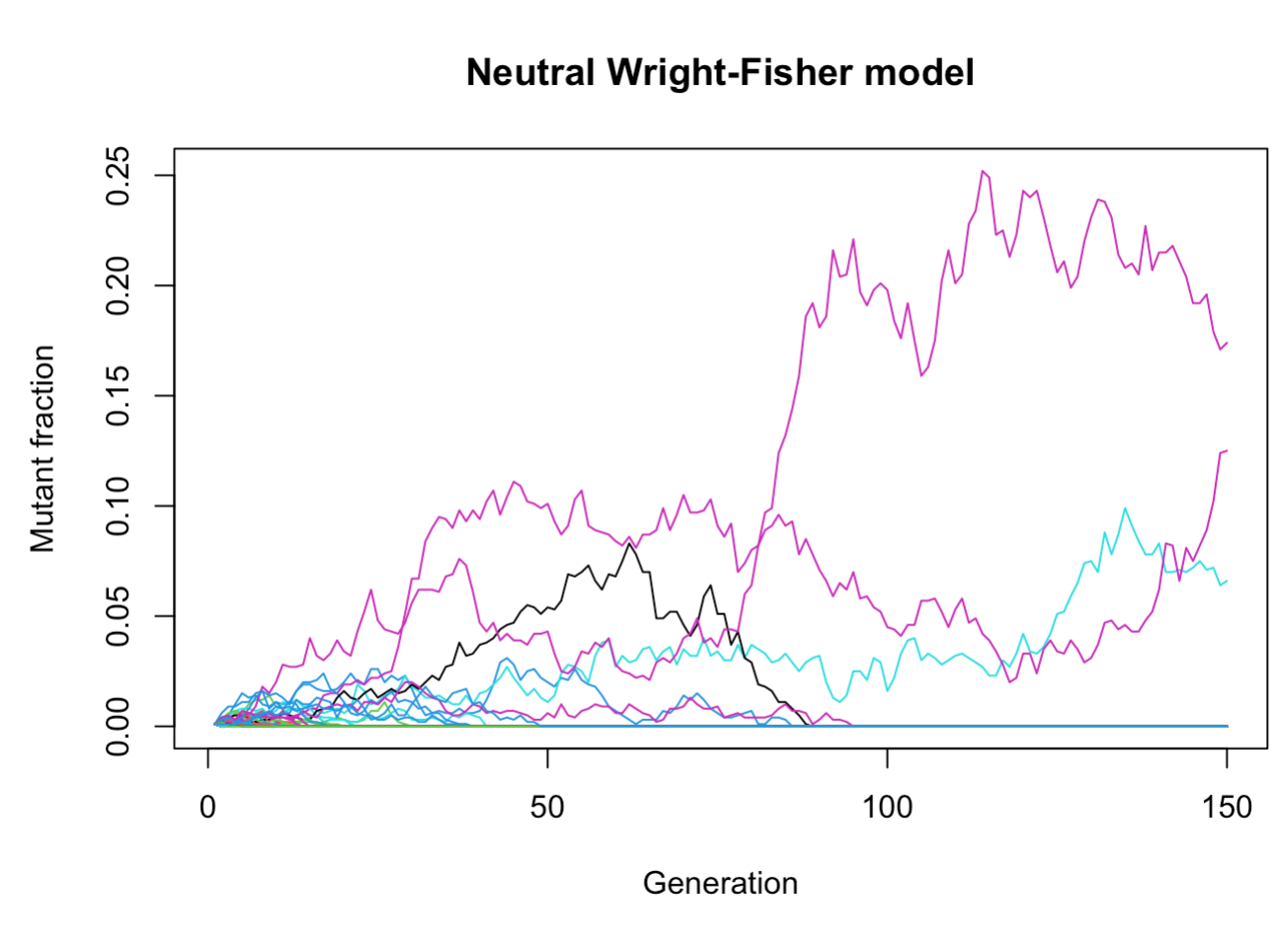
3. With 10 individuals starting with 1 neutral mutant for 150 generations (as above): trajectories show strong fluctuations and end up in either extinction or fixation of the mutant type. Fixation is far less frequent (probability 1/10), but out of 100 realizations we see some fixation events.

```
size=10
mutants_ini=1
n_gen=150
n_rep=100
mutant_frac = matrix(nrow=n_gen,ncol=n_rep)
for(i in 1:n_rep){
  mutant_frac[1,i]=mutants_ini/size
  for(j in 2:n_gen){
    mutant_frac[j,i]=rbinom(1, size, mutant_frac[j-1,i])/size
  }
}
matplot(1:n_gen, mutant_frac, pch=20, type="l", lty="solid", main='Neutral Wrig
```



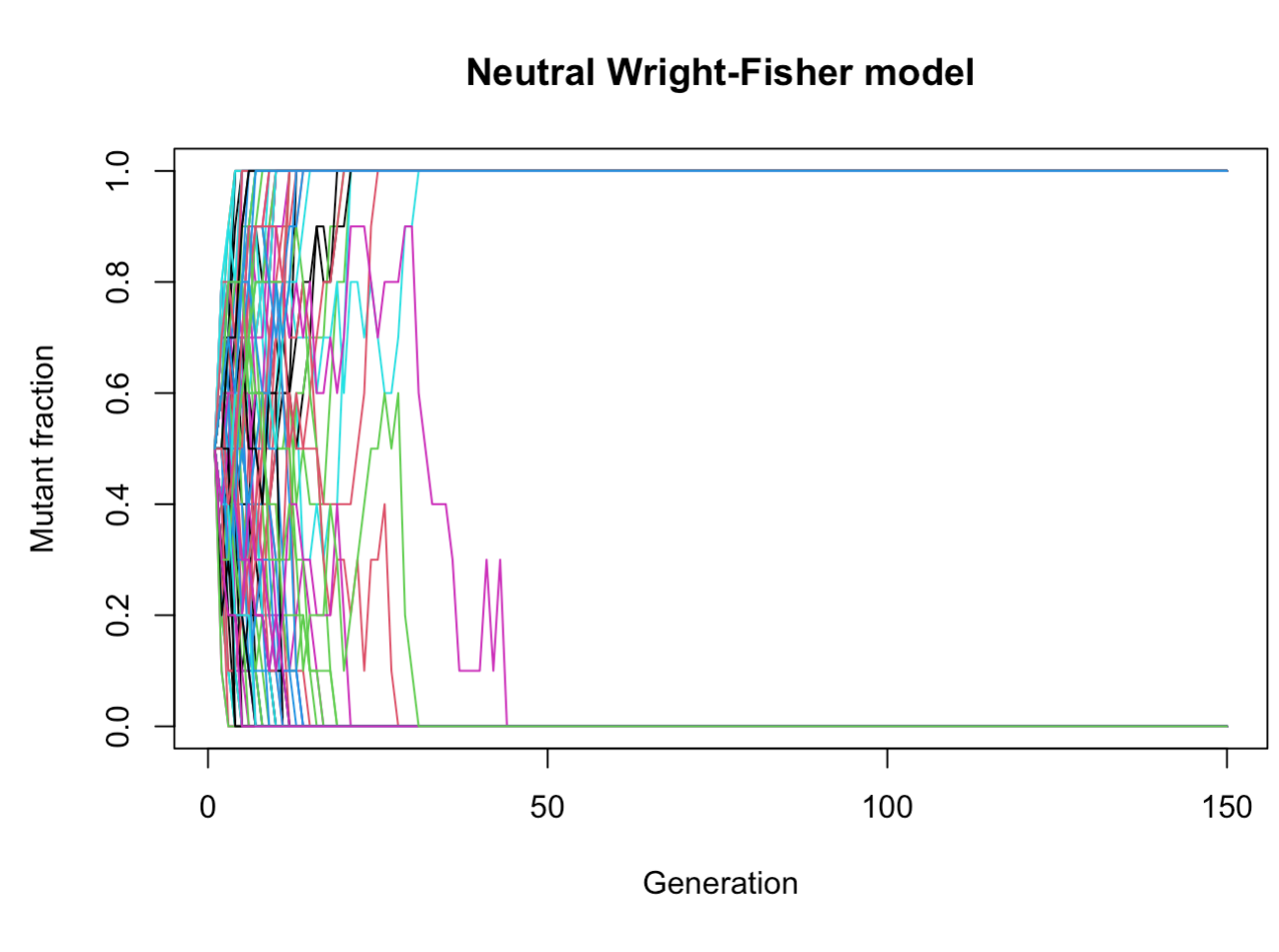
With 1000 individuals starting with 1 neutral mutant for 150 generations, we observe extinctions in the vast majority of cases, usually all of them in 100 realizations. In most trajectories, mutant fraction remains quite small. The probability of fixation is 1/1000 here.

```
size=1000
mutants_ini=1
n_gen=150
n_rep=100
mutant_frac = matrix(nrow=n_gen,ncol=n_rep)
for(i in 1:n_rep){
  mutant_frac[1,i]=mutants_ini/size
  for(j in 2:n_gen){
    mutant_frac[j,i]=rbinom(1, size, mutant_frac[j-1,i])/size
  }
}
matplot(1:n_gen, mutant_frac, pch=20, type="l", lty="solid", main='Neutral Wrig
```



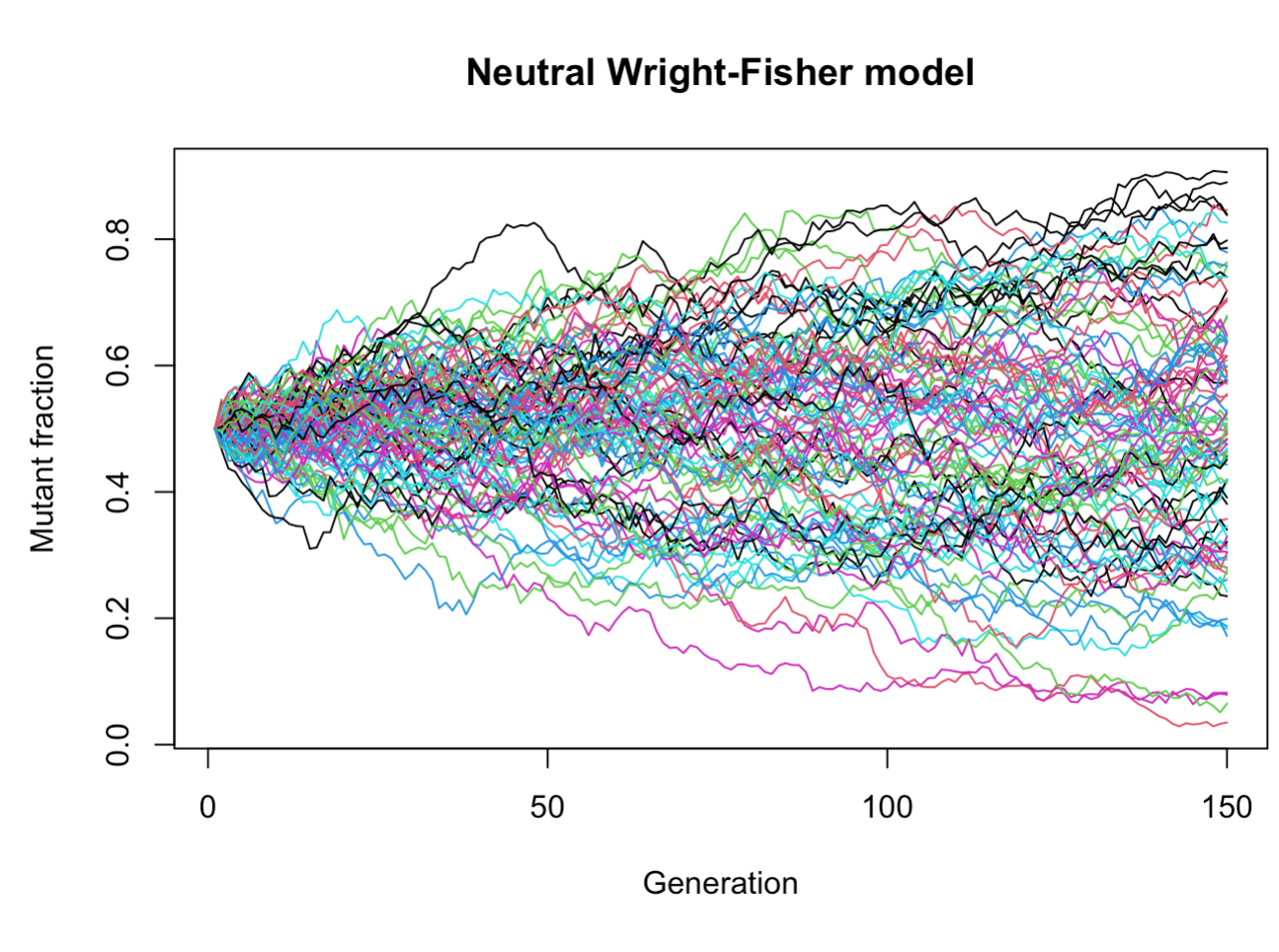
With 10 individuals starting with 5 neutral mutants for 150 generations, we observe strong fluctuations as in the first case, but far more fixation events. Here the probability of fixation is 5/10=0.5.

```
size=10
mutants_ini=5
n_gen=150
n_rep=100
mutant_frac = matrix(nrow=n_gen,ncol=n_rep)
for(i in 1:n_rep){
  mutant_frac[1,i]=mutants_ini/size
  for(j in 2:n_gen){
    mutant_frac[j,i]=rbinom(1, size, mutant_frac[j-1,i])/size
  }
}
matplot(1:n_gen, mutant_frac, pch=20, type="l", lty="solid", main='Neutral Wrig
```



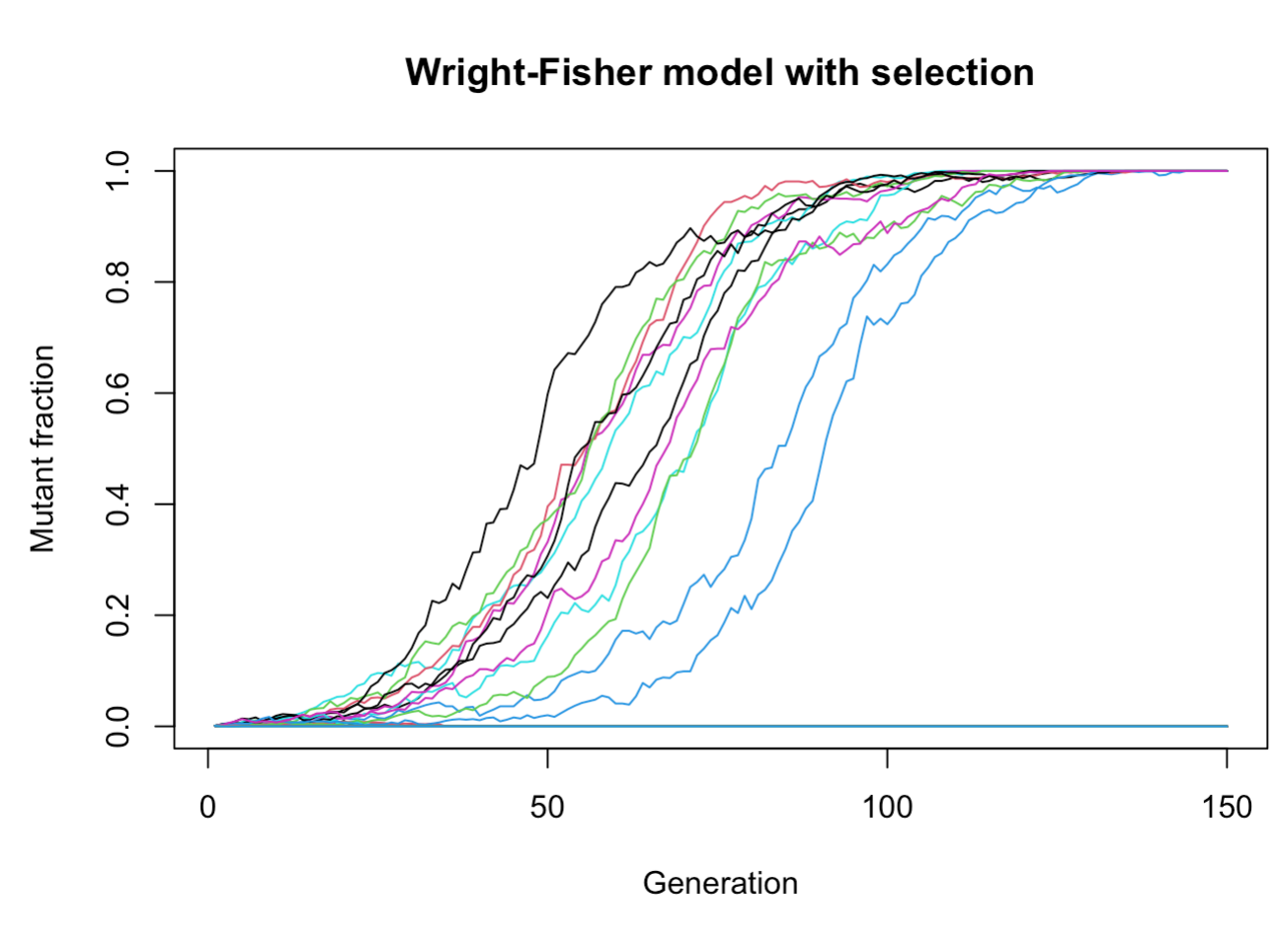
With 1000 individuals starting with 500 neutral mutants for 150 generations, the time of observation is usually not sufficient to obesrve fixation or extinction of the mutant strain. However, increasing this time allows to see fixation or extinction, and all trajectories do end in fixation or extinction of the mutant strain. Here the probability of fixation is again 0.5, but the dynamics are slower because population size is larger. we observe strong fluctuations as in the first case, but far more fixation events.

```
size=1000
mutants_ini=500
n_gen=150
n_rep=100
mutant_frac = matrix(nrow=n_gen,ncol=n_rep)
for(i in 1:n_rep){
  mutant_frac[1,i]=mutants_ini/size
  for(j in 2:n_gen){
    mutant_frac[j,i]=rbinom(1, size, mutant_frac[j-1,i])/size
  }
}
matplot(1:n_gen, mutant_frac, pch=20, type="l", lty="solid", main='Neutral Wrig
```



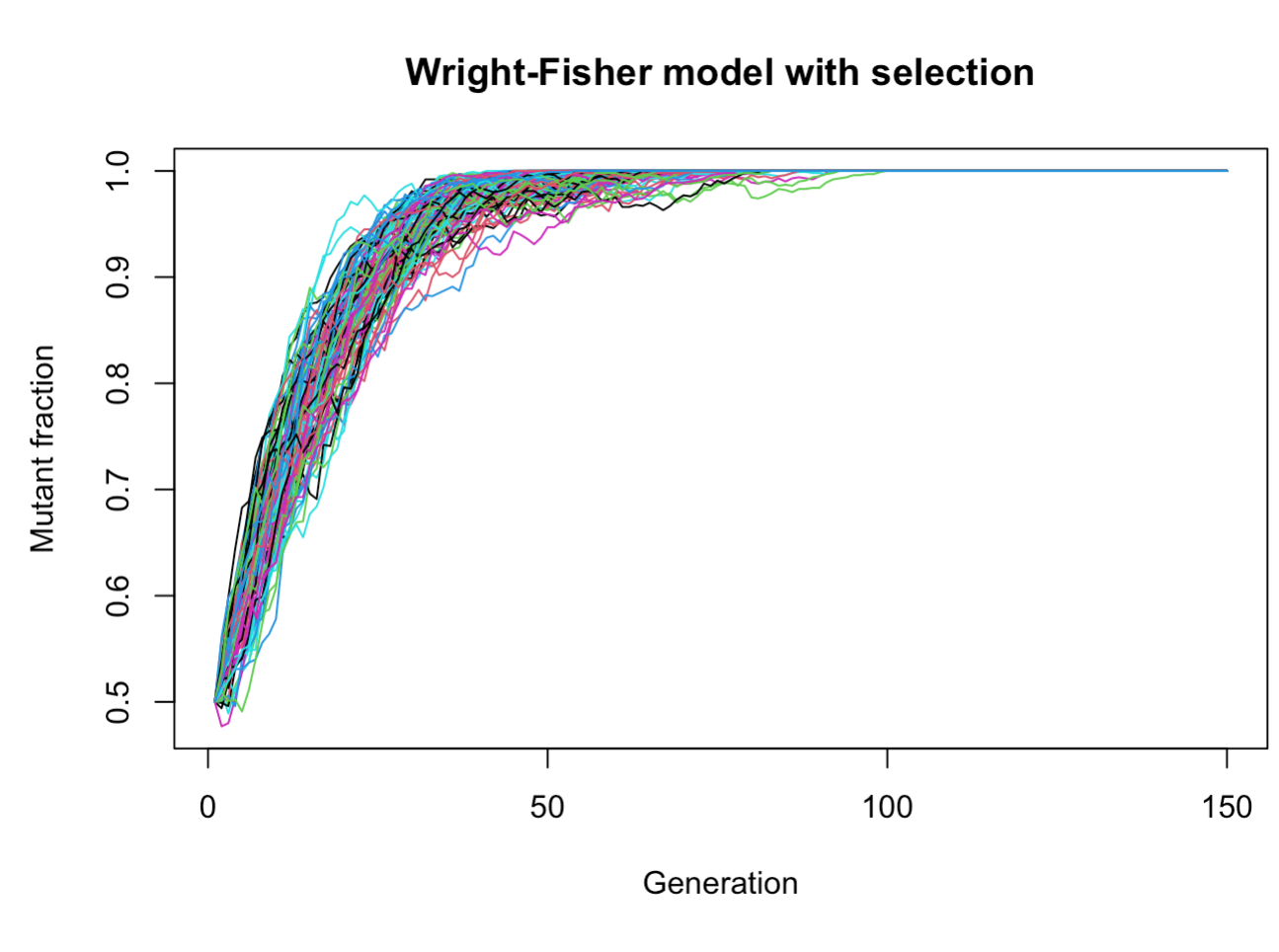
4. To include the effect of natural selection, assuming that the mutants have a relative fitness advantage s compared to the wild-type individuals, we need to change the probability of success (probability of sampling a mutant) in the binomial sampling used to construct generation n+1 from generation n. In the neutral case it was equal to the fraction x of mutants in the population. In the case with selection, fractions have to be reweighted by fitness, so the probability of success becomes (1+s)x/((1+s)x+(1-x))=(1+s)x/(1+sx). This is the point that has to be changed in the code. Note that if s=0 we get back the previous case with a probability of success equal to x. With 1000 individuals starting with 1 mutant with s=0.1 for 150 generations, we observe that most trajectories end in extinction but some give fixation, with noisy but well-defined ascending trajectories for the fraction x of mutants. We are in the regime where s<<1 but s>>1/N, so the expected fixation probability should be close to 2s=0.2, meaning that out of 100 trajectories, we should see around 20 of them that end in fixation.

```
size=1000
mutants_ini=1
s=0.1
n_gen=150
n_rep=100
mutant_frac = matrix(nrow=n_gen,ncol=n_rep)
for(i in 1:n_rep){
  mutant_frac[1,i]=mutants_ini/size
  for(j in 2:n_gen){
    mutant_frac[j,i] = rbinom( 1, size, (1+s)*mutant_frac[j-1,i]/(1+s*mutant_fr
  }
}
matplot(1:n_gen, mutant_frac, pch=20, type="l", lty="solid", main='Wright-Fishe
```



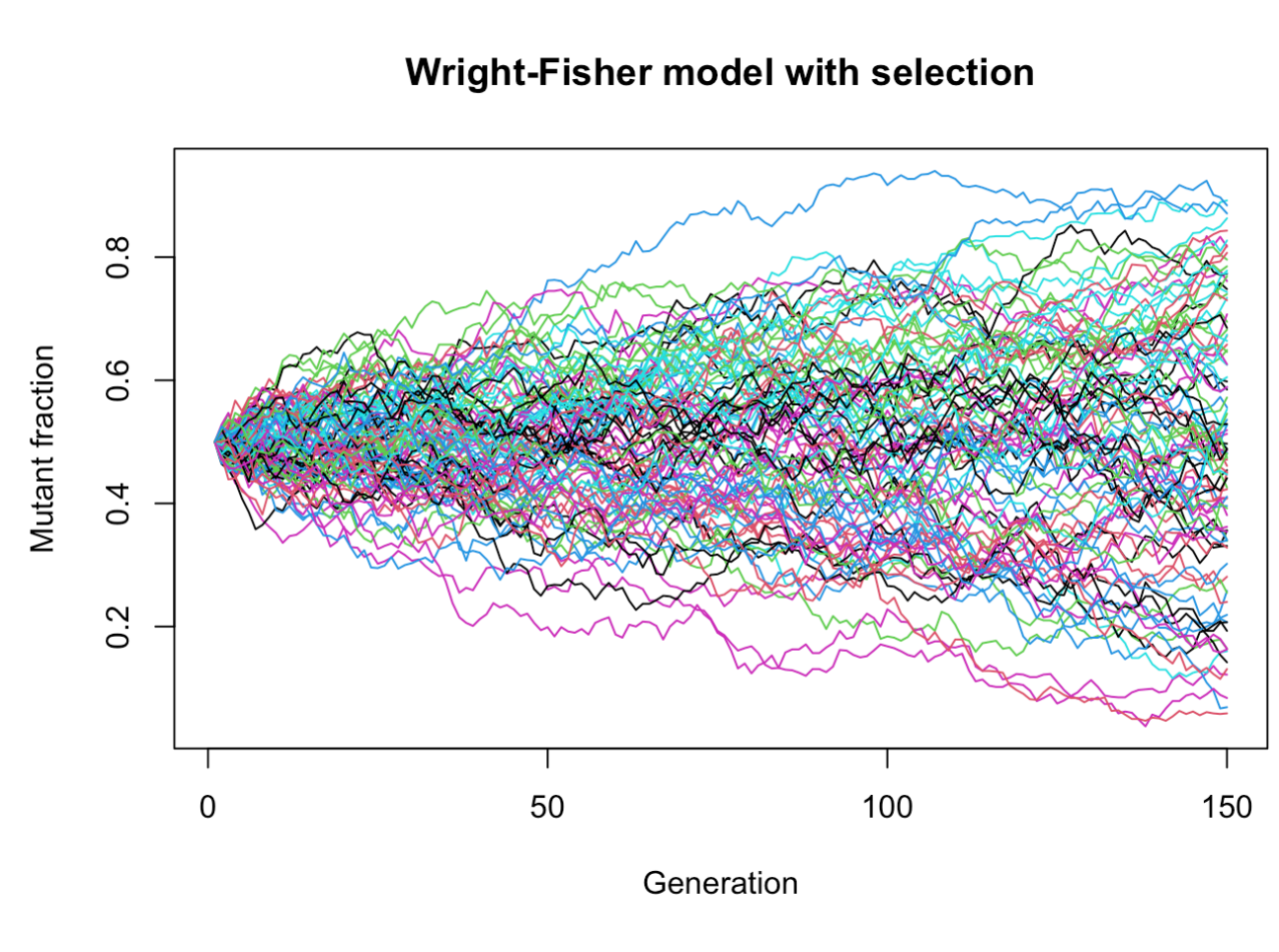
With 1000 individuals starting with 500 neutral mutants with s=0.1 for 150 generations, we observe that all trajectories end in fixation of the mutant. This shows the strong impact of the initial number of mutants. Here s was the same as just before, but the mutant was already abundant from the beginning. When extinction happens for substantially beneficial mutants (s>>1/N) it is usually when their fraction is small. Fluctuations (genetic drift) are then important. If they have reached a certain fraction, these mutants are almost guaranteed to fix.

```
size=1000
mutants_ini=500
s=0.1
n_gen=150
n_rep=100
mutant_frac = matrix(nrow=n_gen,ncol=n_rep)
for(i in 1:n_rep){
  mutant_frac[1,i]=mutants_ini/size
  for(j in 2:n_gen){
    mutant_frac[j,i] = rbinom( 1, size, (1+s)*mutant_frac[j-1,i]/(1+s*mutant_fr
  }
}
matplot(1:n_gen, mutant_frac, pch=20, type="l", lty="solid", main='Wright-Fishe
```



With 1000 individuals starting with 500 neutral mutants with s=0.0001 for 150 generations, we observe a behavior that is very similar to the neutral case with the same population size and initial number of mutants. Indeed here, the selective advantage of mutants is much smaller and we have s<<1/N. The mutant is effectively neutral. One should thus compare s to 1/N to predict whether the system will behave as in the 2nd case studied in this question or as in the 3rd one.

```
size=1000
mutants_ini=500
s=0.0001
n_gen=150
n_rep=100
mutant_frac = matrix(nrow=n_gen,ncol=n_rep)
for(i in 1:n_rep){
  mutant_frac[1,i]=mutants_ini/size
  for(j in 2:n_gen){
    mutant_frac[j,i] = rbinom( 1, size, (1+s)*mutant_frac[j-1,i]/(1+s*mutant_fr
  }
}
matplot(1:n_gen, mutant_frac, pch=20, type="l", lty="solid", main='Wright-Fishe
```



Exercise 3 - Drift, selection and mutation

1. The probability distribution of the number of mutants from type 1 to type 2 at a given generation is binomial with success probability mu and number of trials equal to population size. Indeed, each individual in the population has a probability mu to mutate to another type, and individuals are independent.

2. See code below. The key points are the multinomial sampling to form a new generation in the Wright-Fisher spirit, but generalizing to more than two strains, and then the binomial sampling to model mutations.

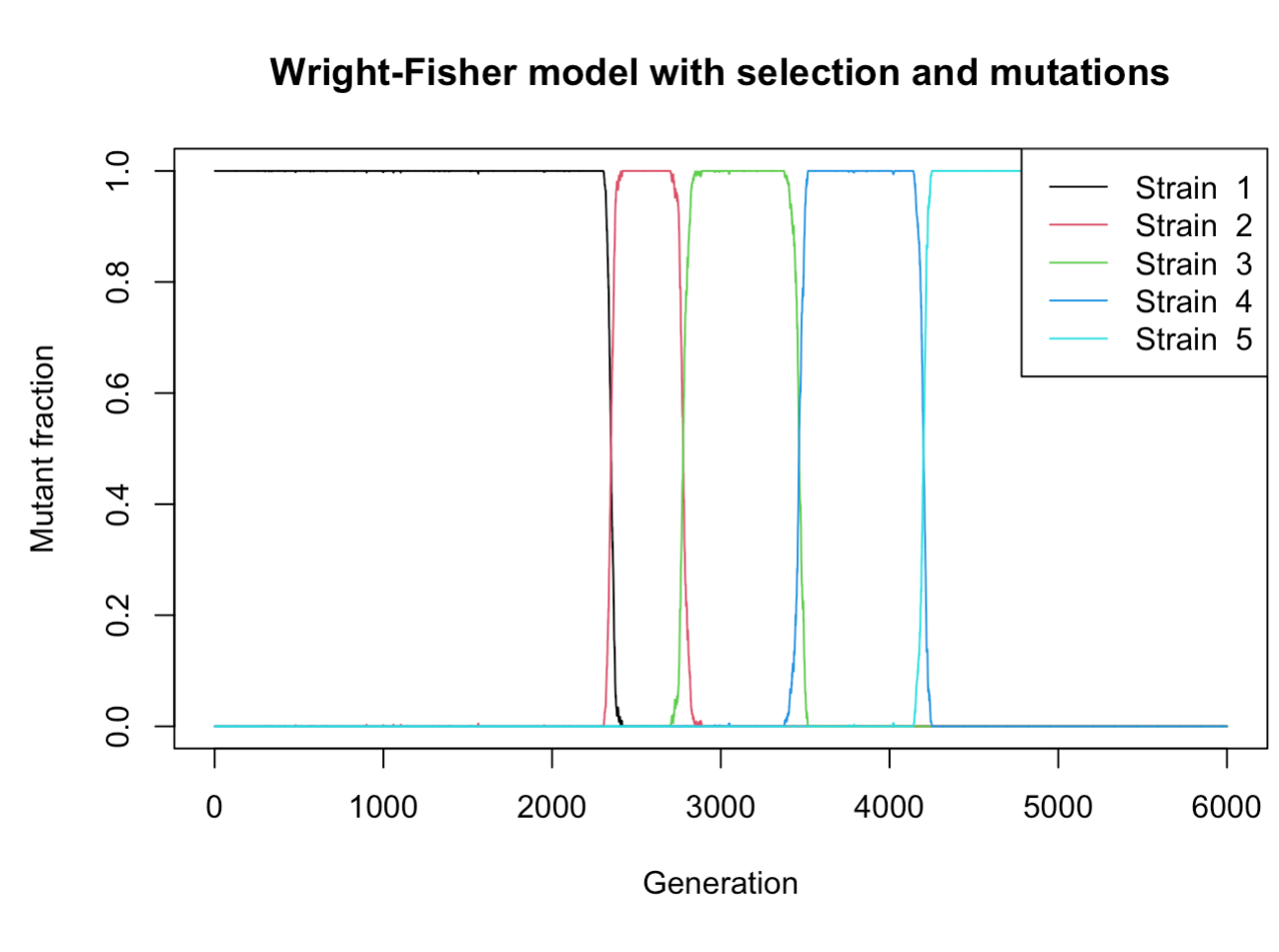
3. In the case where each strain is fitter than the previous one, specifically with fitnesses 1, 1.1, 1.2, 1.3, 1.4, using a population size of 1000 individuals and a mutation probability mu=0.00001, we observe successive fixation events of the different strains. Only one or two strains usually exist in the population in this case - most of the time, a single one, and sometimes two when mutants appear and when they are in the process of taking over. For the population to have such an evolution, where at most two strains are present, mu needs to be small, specifically mu<<1/N.

```
size=1000
n_strains=5
fitness=c(1, 1.1, 1.2, 1.3, 1.4) # or c(1, 1.1, 1, 1.3, 1.4) with mu=0.001 and
mu=0.00001 # or mu=0.001 with n_gen=300
prob = numeric(n_strains)
strain_nmut = numeric(n_strains)
strain_nmut[n_strains] = 0 #last type is assumed not to mutate
strain_num_aftermut = numeric(n_strains)
n_gen=6000
strain_frac = matrix(nrow=n_gen,ncol=n_strains)
strain_frac[1,1]=1

for(i in 2:n_strains){
  strain_frac[1,i]=0
}

for(j in 2:n_gen){
  for(i in 1:n_strains){
    prob[i]=fitness[i]*strain_frac[j-1,i] #in principle we should normalize so
  }
  strain_num_beforemut=t( rmultinom( 1, size, prob ) )
  strain_num_aftermut=strain_num_beforemut
  for(i in 1:(n_strains-1)){
    strain_nmut[i]=rbinom( 1, strain_num_beforemut[i], mu )
  }
  for(i in 1:(n_strains-1)){ #last type is assumed not to mutate
    strain_num_aftermut[i]=strain_num_aftermut[i]-strain_nmut[i]
    strain_num_aftermut[i+1]=strain_num_aftermut[i+1]+strain_nmut[i]
  }
  strain_frac[j, ] = strain_num_aftermut / size
}

matplot(1:n_gen, strain_frac, pch=20, type="l", lty="solid", main='Wright-Fishe
labels = sprintf('Strain %2d', 1:n_strains)
legend("topright", lty="solid", col=1:n_strains, labels)
```



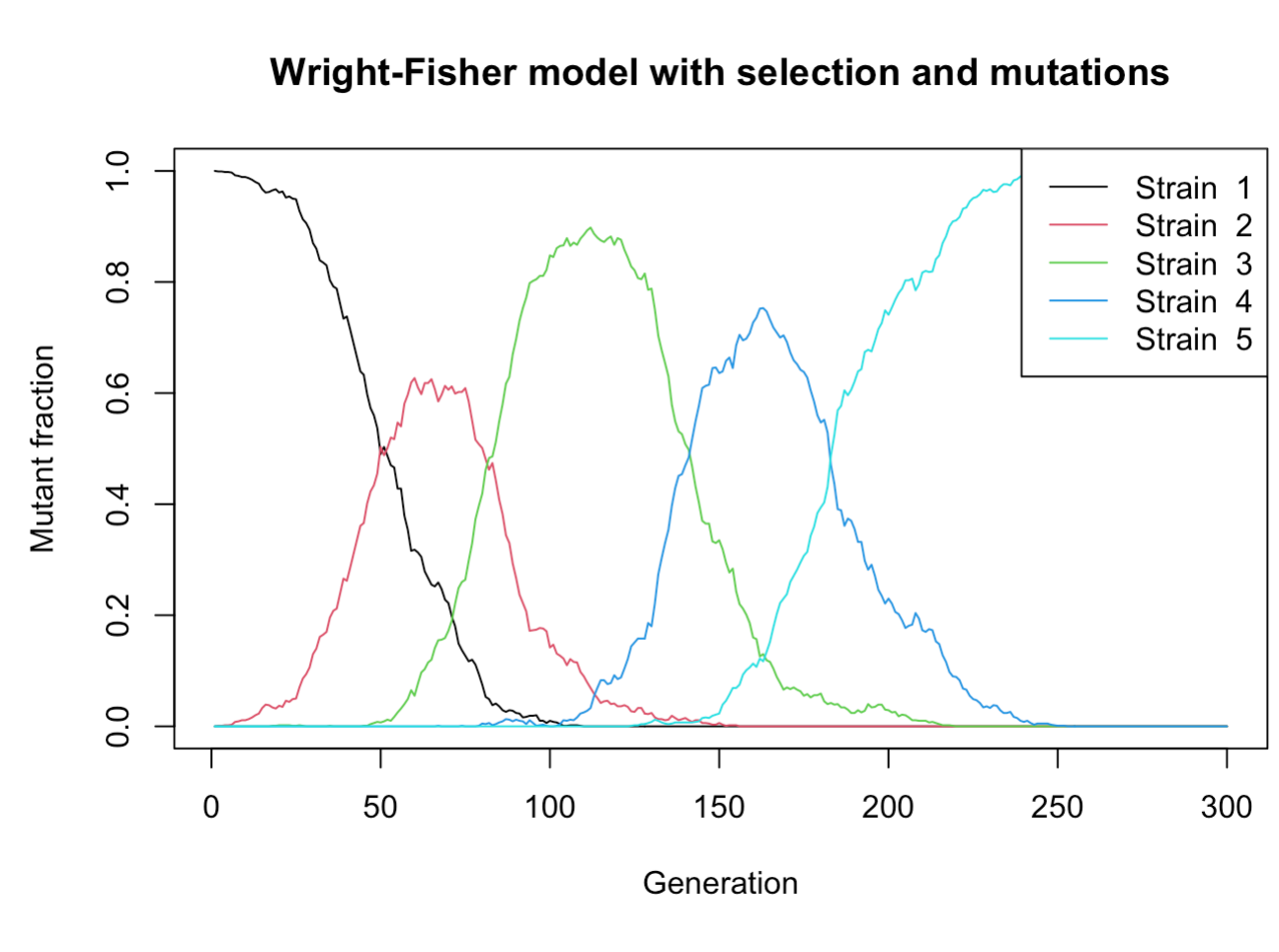
4. Using a mutation probability mu=0.001, mutations are much more frequent and we are no longer in the regime mu<<1/N. Thus, mutants of strain n+2 usually appear from strain n+1 while strain n+1 is taking over and in the process of replacing strain n. This means that more than two strains usually coexist, before the last strain takes over. This is called the clonal interference regime.

```
size=1000
n_strains=5
fitness=c(1, 1.1, 1.2, 1.3, 1.4) # or c(1, 1.1, 1, 1.3, 1.4) with mu=0.001 and
mu=0.001
prob = numeric(n_strains)
strain_nmut = numeric(n_strains)
strain_nmut[n_strains] = 0 #last type is assumed not to mutate
strain_num_aftermut = numeric(n_strains)
n_gen=300
strain_frac = matrix(nrow=n_gen,ncol=n_strains)
strain_frac[1,1]=1

for(i in 2:n_strains){
  strain_frac[1,i]=0
}

for(j in 2:n_gen){
  for(i in 1:n_strains){
    prob[i]=fitness[i]*strain_frac[j-1,i] #in principle we should normalize so
  }
  strain_num_beforemut=t( rmultinom( 1, size, prob ) )
  strain_num_aftermut=strain_num_beforemut
  for(i in 1:(n_strains-1)){
    strain_nmut[i]=rbinom( 1, strain_num_beforemut[i], mu )
  }
  for(i in 1:(n_strains-1)){ #last type is assumed not to mutate
    strain_num_aftermut[i]=strain_num_aftermut[i]-strain_nmut[i]
    strain_num_aftermut[i+1]=strain_num_aftermut[i+1]+strain_nmut[i]
  }
  strain_frac[j, ] = strain_num_aftermut / size
}

matplot(1:n_gen, strain_frac, pch=20, type="l", lty="solid", main='Wright-Fishe
labels = sprintf('Strain %2d', 1:n_strains)
legend("topright", lty="solid", col=1:n_strains, labels)
```



5. Assuming that the fitnesses of the successive strains are 1, 1.1, 1, 1.3, 1.4, the step going from the second to the third strain is special because the third strain is less fit than the second one. So far, we only considered beneficial mutations, but this particular mutation is deleterious. Using a mutation probability mu=0.0005, and n_gen=5000 generations, we observe that the third strain does not take over or reach high fractions, which is very different from the previous cases. The fourth strain emerges from a small minority of individuals of the third strain. This process takes time, and as a result, the second strain dominates for a long time. If mu was much smaller, this process would become much slower. Remark: In fact, below a certain mu, we would usually have to wait until the deleterious mutant (strain 3) fixes to then get a fixation of strain 4.

```
size=1000
n_strains=5
fitness=c(1, 1.1, 1, 1.3, 1.4)
mu=0.0005
prob = numeric(n_strains)
strain_nmut = numeric(n_strains)
strain_nmut[n_strains] = 0 #last type is assumed not to mutate
strain_num_aftermut = numeric(n_strains)
n_gen=5000
strain_frac = matrix(nrow=n_gen,ncol=n_strains)
strain_frac[1,1]=1

for(i in 2:n_strains){
  strain_frac[1,i]=0
}

for(j in 2:n_gen){
  for(i in 1:n_strains){
    prob[i]=fitness[i]*strain_frac[j-1,i] #in principle we should normalize so
  }
  strain_num_beforemut=t( rmultinom( 1, size, prob ) )
  strain_num_aftermut=strain_num_beforemut
  for(i in 1:(n_strains-1)){
    strain_nmut[i]=rbinom( 1, strain_num_beforemut[i], mu )
  }
  for(i in 1:(n_strains-1)){ #last type is assumed not to mutate
    strain_num_aftermut[i]=strain_num_aftermut[i]-strain_nmut[i]
    strain_num_aftermut[i+1]=strain_num_aftermut[i+1]+strain_nmut[i]
  }
  strain_frac[j, ] = strain_num_aftermut / size
}

matplot(1:n_gen, strain_frac, pch=20, type="l", lty="solid", main='Wright-Fishe
labels = sprintf('Strain %2d', 1:n_strains)
legend("topright", lty="solid", col=1:n_strains, labels)
```