

Classical models of group selection

Haldane

(according to “The Causes of Evolution”, appendix “Socially valuable but individually disadvantageous characters”, 1932)

populations grow, within population competition, and split if they get too big.

We will use the same dynamics within the group as we did before: (Haldane’s model is diploid, ours is haploid)

populations grow, within population competition, and split if they get too big.

We will use the same dynamics within the group as we did before:

```
> f=1:2; f[1]=1; f[2]=1.05
> f
[1] 1.00 1.05
> g=sample(1:2,10,rep=T)
> g
[1] 1 2 1 2 2 1 1 2 2 2
> N=length(g)
individuals=1:N
fitnesses=f[g]
parents=sample(individuals,N,replace=TRUE,prob=fitnesses/sum(fitnesses) )
g.offsprings=g[parents]
>
> g.offsprings
[1] 1 2 2 2 2 1 1 2 1 2
>
>
```

And, again we will let the population grow according to a benefit gained from altruists:

```
> b=0.05
> N=length(g)
Np=N*(1+sum(g==1)/N*b)
> N
> Np
[1] 10
>
```

But, now we will have several populations. This is done by using a list of populations

```
> replicate(3,1:4,simplify=F)
[1] 10.2
> l=replicate(3,1:4,simplify=F)
[[1]]
[1] 1 2 3 4
```

```

[[2]]
[1] 1 2 3 4

[[3]]
[1] 1 2 3 4
> l
[[1]]
[1] 1 2 3 4

[[2]]
[1] 1 2 3 4

[[3]]
[1] 1 2 3 4
> l[[1]]
[1] 1 2 3 4
> l[[2]]
[1] 1 2 3 4
>

```

We created a list that has 3 elements, each of which is a list. This is very useful for dealing with populations:

```

> P=replicate(10,sample(1:2,10,rep=T),simp=F)
> P
[[1]]
[1] 1 1 1 2 2 2 1 1 2 2

[[2]]
[1] 2 2 2 2 2 2 1 1 2

[[3]]
[1] 1 1 1 2 2 1 1 1 1 1

[[4]]
[1] 2 2 2 1 1 1 2 2 2 1

[[5]]
[1] 2 2 2 2 2 2 1 1 1 2

[[6]]
[1] 2 1 2 1 2 2 1 1 2 1

[[7]]
[1] 1 1 1 1 1 2 2 2 2 2

[[8]]
[1] 1 2 2 1 1 2 2 2 2 2

[[9]]
[1] 1 2 2 2 2 1 2 2 1 1

```

```
[[10]]
 [1] 2 1 1 1 2 1 2 2 2 1
>
```

Just as with vectors, we can access parts of lists:

```
> P[1:2]
[[1]]
 [1] 1 1 1 2 2 2 1 1 2 2

[[2]]
 [1] 2 2 2 2 2 2 2 1 1 2
> P[2:4]
[[1]]
 [1] 2 2 2 2 2 2 2 1 1 2

[[2]]
 [1] 1 1 1 2 2 1 1 1 1 1

[[3]]
 [1] 2 2 2 1 1 1 2 2 2 1
>
```

Notice the confusing difference:

```
> P[1:2]
[[1]]
 [1] 1 1 1 2 2 2 1 1 2 2

[[2]]
 [1] 2 2 2 2 2 2 2 1 1 2
> P[1]
[[1]]
 [1] 1 1 1 2 2 2 1 1 2 2
>
```

The result in both cases is a list, the second one of length 1.

```
> P[[1]]
 [1] 1 1 1 2 2 2 1 1 2 2
>
```

The the result now is a vector, it is just an element of the list!

We can now just do operations on all elements of the list:

```
> for(g in P) {
  print(g)
}
+ + b=0.3; f=1:2; f[1]=1; f[2]=1.05
 [1] 1 1 1 2 2 2 1 1 2 2
 [1] 2 2 2 2 2 2 2 1 1 2
```

```

[1] 1 1 1 2 2 1 1 1 1 1
[1] 2 2 2 1 1 1 2 2 2 1
[1] 2 2 2 2 2 2 1 1 1 2
[1] 2 1 2 1 2 2 1 1 2 1
[1] 1 1 1 1 1 2 2 2 2 2
[1] 1 2 2 1 1 2 2 2 2 2
[1] 1 2 2 2 2 1 2 2 1 1
[1] 2 1 1 1 2 1 2 2 2 1
> for(g in P) {
  N=length(g)
  fitnesses=f[g]
  Np=N*(1+sum(g==1)/N*b)
  parents=sample(individuals,Np,replace=T,prob=fitnesses/sum(fitnesses) )
  g.offsprings=g[parents]
  print(g.offsprings)
}
+ + + + + + + [1] 1 2 1 1 2 1 1 1 2 1 2
[1] 2 2 2 2 2 2 2 2 2 2
[1] 2 1 1 2 2 1 1 1 1 1 1 1
[1] 1 1 1 1 2 1 2 1 2 1 1
[1] 2 2 2 2 2 1 2 1 2 2
[1] 1 1 1 2 2 2 1 1 1 2 2
[1] 2 1 1 1 2 1 2 2 2 1 2
[1] 2 2 2 2 2 1 2 2 1 2
[1] 2 2 2 1 1 2 1 2 2 1 1
[1] 1 2 2 2 1 1 1 2 1 1 2
>
>

```

Now we only need 2 more things:

We need to remember the new populations - till now we only printed them out, and we need to split a population if it is big enough.

```

> new.P=list()
> new.P[[length(new.P)+1]]=1:3
> new.P
[[1]]
[1] 1 2 3
> new.P[[length(new.P)+1]]=1:3
> new.P
[[1]]
[1] 1 2 3

```

```
[[2]]
[1] 1 2 3
```

>

So, now we can produce the new population:

```
> new.P=list()
for(g in P) {
  N=length(g)
  fitnesses=f[g]
  Np=N*(1+sum(g==1)/N*b)
  parents=sample(individuals,Np,replace=T,prob=fitnesses/sum(fitnesses) )
  g.offsprings=g[parents]
  new.P[[length(new.P)+1]]=g.offsprings
}
```

> + + + + + + +

> new.P

```
[[1]]
[1] 1 1 1 2 1 2 1 2 1 1 1

[[2]]
[1] 1 2 1 1 2 2 1 2 2 2

[[3]]
[1] 2 1 1 1 1 1 2 1 2 1 1 1

[[4]]
[1] 1 1 1 1 1 2 2 2 1 1 2

[[5]]
[1] 1 1 2 2 1 2 1 2 2 1

[[6]]
[1] 1 2 2 1 1 2 2 2 2 1

[[7]]
[1] 2 2 2 2 2 1 1 2 1 2 1

[[8]]
[1] 2 1 2 2 2 2 1 2 1 2

[[9]]
[1] 2 2 2 2 1 2 2 2 2 1

[[10]]
[1] 1 1 1 1 1 1 1 2 2 2 1
```

>

>

>

Finally the split:

> g

```
[1] 2 1 1 1 2 1 2 2 2 1
```



```

[[3]]
[1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

[[4]]
[1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

[[5]]
[1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

[[6]]
[1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

[[7]]
[1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

[[8]]
[1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

[[9]]
[1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

[[10]]
>
[1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
>

```

The altruists have won!

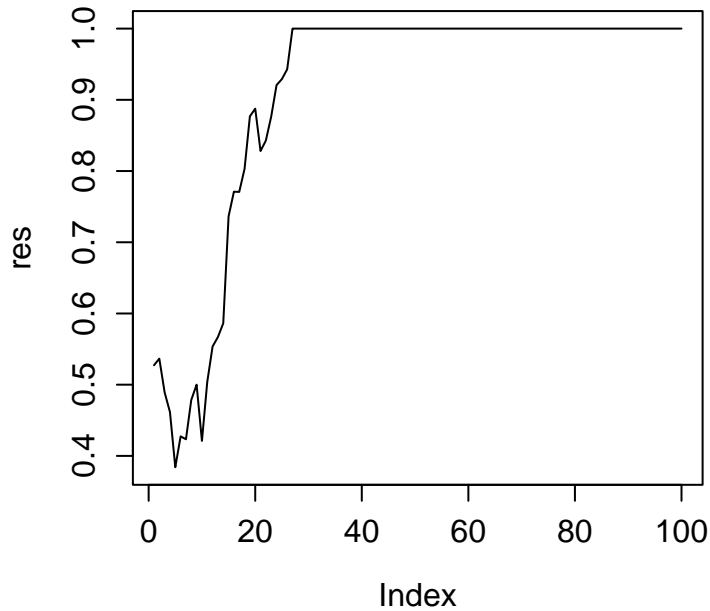
Let us see what happened:

```

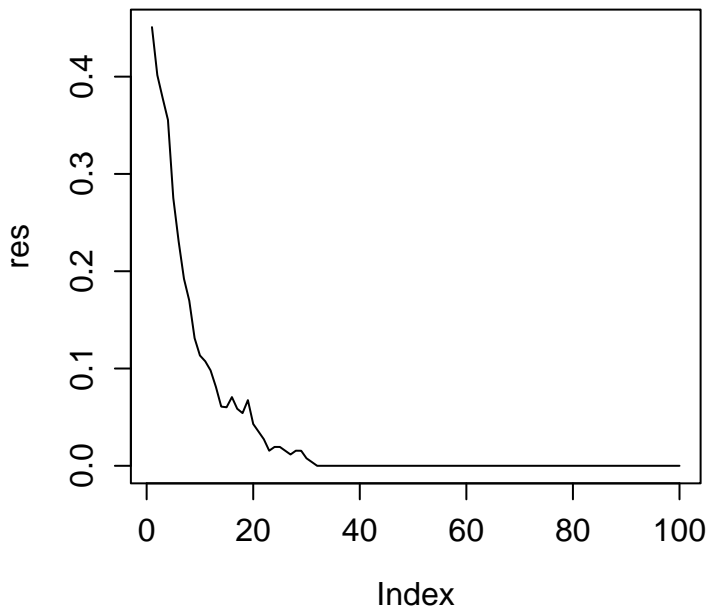
> P=replicate(10,sample(1:2,10,rep=T),simp=F)
> b=0.3; f=1:2; f[1]=1; f[2]=1.2
> res=1:100
> for( generations in 1:100) {
  new.P=list()
  for(g in P) {
    N=length(g)
    fitnesses=f[g]
    Np=N*(1+sum(g==1)/N*b)
    individuals=1:N
    parents=sample(individuals,Np,replace=T,prob=fitnesses/sum(fitnesses) )
    g.offsprings=g[parents]
    Np=length(g.offsprings)
    if( Np > 20 ) {
      m=Np/2
      new.P[[length(new.P)+1]]=g.offsprings[1:m]
      new.P[[length(new.P)+1]]=g.offsprings[(m+1):Np]
    } else {
      new.P[[length(new.P)+1]]=g.offsprings
    }
  }
  P = sample(new.P,10)
  res[generations]=sum(unlist(P)==1)/length(unlist(P))
}

```

```
+
> plot(res,type="l");v()
```



```
>
> P=replicate(10,sample(1:2,20,rep=T),simp=F)
> b=0.3; f=1:2; f[1]=1; f[2]=1.2
> res=1:100
> for( generations in 1:100) {
  new.P=list()
  for(g in P) {
    N=length(g)
    fitnesses=f[g]
    Np=N*(1+sum(g==1)/N*b)
    individuals=1:N
    parents=sample(individuals,Np,replace=T,prob=fitnesses/sum(fitnesses) )
    g.offsprings=g[parents]
    Np=length(g.offsprings)
    if( Np > 40 ) {
      m=Np/2
      new.P[[length(new.P)+1]]=g.offsprings[1:m]
      new.P[[length(new.P)+1]]=g.offsprings[(m+1):Np]
    } else {
      new.P[[length(new.P)+1]]=g.offsprings
    }
  }
  P = sample(new.P,10)
  res[generations]=sum(unlist(P)==1)/length(unlist(P))
}
+
> plot(res,type="l");v()
```



> P.N=100

>

We see that with a population size of 20, you need many populations for the altruists to win.

Why is this?

In every population, other than one of only altruists, selfish individuals win.

If at some point, an all-altruist population is produced, it takes over with a high chance.

Homework

start with a single population with a fixed number of altruists and selfish individuals (for example 17 altruists and 3 selfish.) Evolve that one population for several generations, without sub-sampling populations, and look at the resulting distribution of populations.

How often do you get 18-2, 19-1, 20-0?