

## Excercise 2

Help: you can write a series of R commands into a file, and then execute them all in a row by doing

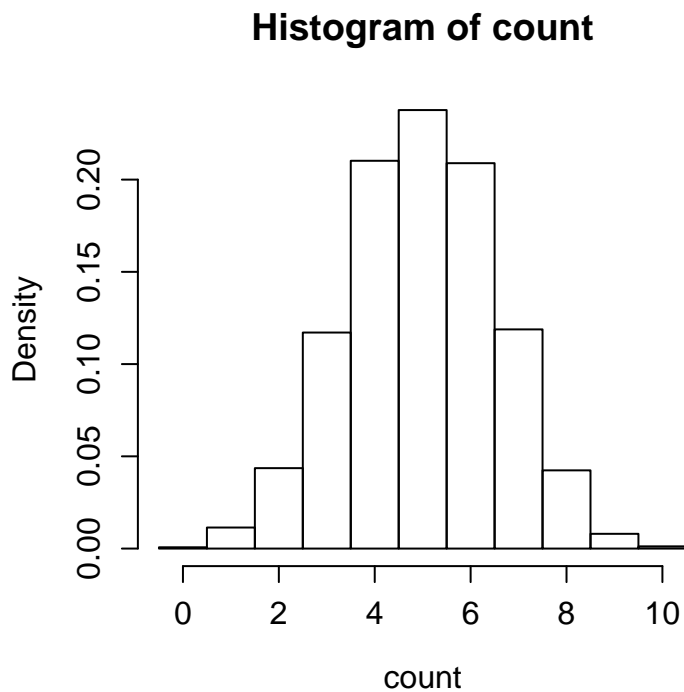
```
> source("filename")
```

1a. By simulating 10000 times flipping a coin 10 times, make a table of the probability to have heads show up 0 times, 1 time, twice, and so on.

```
> flip=matrix(sample(0:1,10000*10,rep=T),10000,10)
> count=apply(flip,1,sum)
> table(count)/length(count)
```

```
count
  0    1    2    3    4    5    6    7    8    9   10
0.0007 0.0114 0.0436 0.1171 0.2102 0.2377 0.2089 0.1188 0.0424 0.0080 0.0012
```

```
> hist(count,breaks=-0.5:10.5,prob=T);v()
```



```
>
```

1b. If someone flipped a coin, and got 1 head, and 9 tails, we would like to test the hypothesis that the coin is unbiased. We will do this by calculating the probability that an unbiased coin flipped 10 times gives 0 or 1 heads, or 0 or 1 tails. Calculate this probability.

```
> Tab=table(count)/length(count)
```

```

> Tab
      count
      0     1     2     3     4     5     6     7     8     9    10
0.0007 0.0114 0.0436 0.1171 0.2102 0.2377 0.2089 0.1188 0.0424 0.0080 0.0012
> sum(count <= 1 | count >= 10-1)/length(count)
[1] 0.0213

```

>

Notice that the '|' symbol means 'or' and the '&' symbol means 'and'.

1c. How many heads/tails out of 10 would cause one to reject the hypothesis that the coin is unbiased at the 5% level? That means, how many heads/tails give a probability that is lower than 5% as calculated above.

```

> sapply(0:5,function(i) sum(count <= i | count >= 10-i)/length(count))
[1] 0.0019 0.0213 0.1073 0.3432 0.7623 1.0000

```

>

At a 5% level, we would reject 1:9 but not 2:8.

1d. With the command `sample(0:1, 10, prob=c(0.3,0.7))` you can simulate a coin flip of a biased coin. Out of 1000 coin flips of length 10 that are biased as above, how many come up significant at the 5% level? I.e. how often do you get as low a number as you calculated in 1c or worse? How many come up significant for an unbiased coin? How about `c(0.1,0.9)` ?

```

> bflip=matrix( sample(0:1, 10*1000, rep=T, prob=c(0.3,0.7) ), 1000, 10)
> bcount=apply(bflip,1,sum)

```

Since we reject at the 5% level 0:10, 1:9, 9:1, and 10:0, we count how often these occur:

```

> sum(bcount <= 1 | bcount >= 10-1)/length(bcount)
[1] 0.144

```

>

So in 14% of the cases we recognize the biased coin as biased. Notice that an unbiased coin we would mark as biased in 5% of the cases.

```

> bflip=matrix( sample(0:1, 10*1000, rep=T, prob=c(0.1,0.9) ), 1000, 10)
> bcount=apply(bflip,1,sum)
> sum(bcount <= 1 | bcount >= 10-1)/length(bcount)

```

```
[1] 0.739
```

>

In this case, we recognize the coin as biased in 74% of the cases.

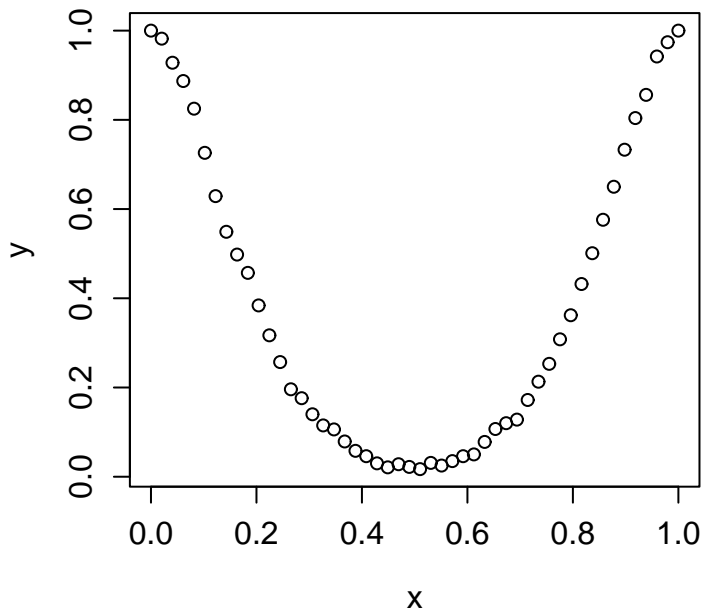
We can try to do a graph of this:

```

> recog=function(p) {
  bflip=matrix( sample(0:1, 10*1000, rep=T, prob=c(p,1-p) ), 1000, 10)
  bcount=apply(bflip,1,sum)
  sum(bcount <= 1 | bcount >= 10-1)/length(bcount)
}
+ + + + > recog(0.1)
[1] 0.74
> x=seq(0,1,length=50)
> y=sapply(x,recog)
> plot(x,y,main="Chance of recognition of a biased coin");v()

```

## Chance of recognition of a biased coin



>

Notice that we had to use `y=sapply(x,recog)` and not `y=recog(x)`, because we wrote `recog` so that it works on single numbers, and not on vectors. It would be better to write it to work on vectors.

>

2. If the sex ratio at birth is 50%, but people adopt the following strategy: have children until you have one son, and then stop, unless you have 12 daughters (in which case you stop, too). What would be the average sex ratio in the population? (Calculate by simulation!)

```

> kids=matrix(sample(0:1,10000*12,rep=T),10000,12)
> daughters=apply(kids,1,function(x){ if( sum(x)==0 ) 12 else min( which( x==1 ) )-1} )
> sons=last.daughter<12
> table(sons,daughters)

```

daughters

```

sons    0    1    2    3    4    5    6    7    8    9    10   11   12
FALSE   0    0    0    0    0    0    0    0    0    0    0    0    3
TRUE  5053 2411 1235  640  321  168   90   39   21   6   10   3    0

```

```

> sum(sons)/sum(daughters)

[1] 0.9869681

```

We see that there seem to be equal numbers of sons and daughters. Let us test this:

```

> prop.test(sum(sons),n=sum(sons)+sum(daughters),p=0.5)

```

```

1-sample proportions test with continuity correction

```

```

data:  sum(sons) out of sum(sons) + sum(daughters), null probability 0.5
X-squared = 0.8527, df = 1, p-value = 0.3558
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
 0.4897895 0.5036531
sample estimates:
      p
0.4967207

```

```

>

```

We don't see a significant difference from equal probabilities.

3a. As we did in Wednesday's exercise, start with a population of 100 individuals. We will assume that they all have different alleles, so that their alleles are 1:100. Resample from the population 300 times - how many alleles are still present from the original population?

```

> pop=1:100
> new.gen=function(pop){sample(pop,rep=T)}
> for(i in 1:300){ pop=new.gen(pop) }
> table(pop)

```

```

pop
 58
100

```

```

> length(unique(pop))

[1] 1

```

```

>

```

Only one genotype survived. `unique(x)` shows how many different numbers (or strings) are in `x`.

Here is a cool way to display this:

```

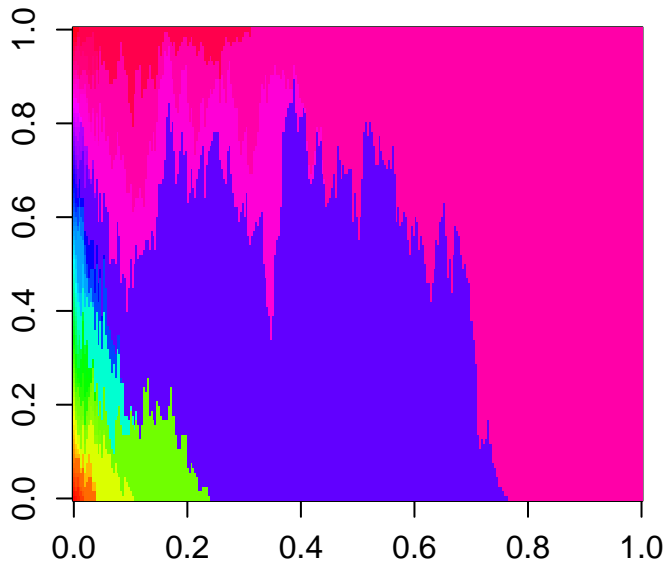
> pop=1:100

```

```
> gens=matrix(0,300,100);gens[1,]=pop
> for(i in 1:299){gens[i+1,]=new.gen(gens[i,])}
>
```

Now we have the whole timeline. Now lets sort it:

```
> gens=t(apply(gens,1,sort))
> image(gens,col=rainbow(100));v()
```



>

Cool, isn't it? We see one genotype spreading over the population. By sorting I put all the individuals of the same type near one another. `image()` takes a matrix and makes a color image out of it.

3b. Now represent the population as a matrix, so that the rows are individuals, and the columns are loci (i.e. individuals are haploids). create a new population by choosing a mother and a father for each individual, and then choosing the alleles randomly from the father and the mother for each locus. Start with a population of 20 individuals with a genome length of 100. Alleles from how many individuals are present after 300 generations?

```
> pop=matrix(1:20,20,100)
> pop[1:3,1:12]
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]	[,11]	[,12]
[1,]	1	1	1	1	1	1	1	1	1	1	1	1
[2,]	2	2	2	2	2	2	2	2	2	2	2	2
[3,]	3	3	3	3	3	3	3	3	3	3	3	3

>

We created the population. Each row is an individual, each column is a locus. At each locus each individual is distinguishable. We want to see from how many individuals do we still see alleles at the end.

```
> mom.i=sample(1:20,rep=T)
> dad.i=sample(1:20,rep=T)
> mom=t( sapply( mom.i, function(i) pop[i,] ) )
> dad=t( sapply( dad.i, function(i) pop[i,] ) )
>
```

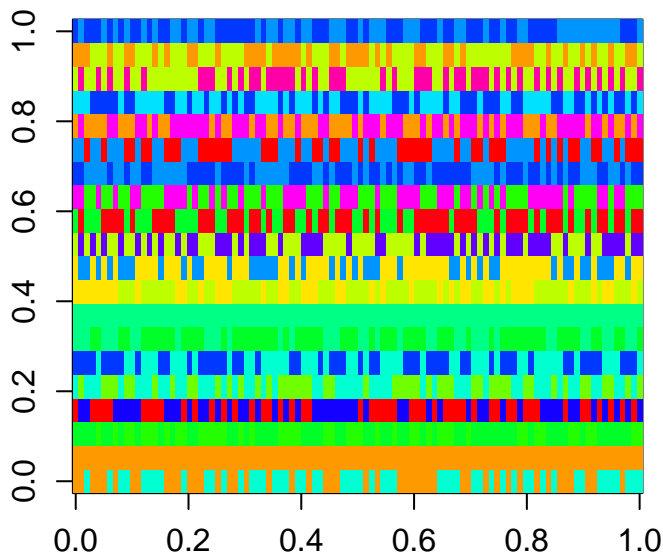
Now we have a matrix with the genotypes of the mothers, and a matrix with the genotypes of the fathers.

To recombine we will randomly chose either from one, or from the other:

```
> rec=matrix(sample(c(T,F), 20*100,rep=T),20,100)
> next.gen=mom
> next.gen[rec]=dad[rec]
> next.gen[1:3,1:12]
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]	[,11]	[,12]
[1,]	3	3	10	3	3	3	10	3	10	10	3	3
[2,]	3	3	3	3	3	3	3	3	3	3	3	3
[3,]	8	8	8	8	7	8	8	7	8	7	7	8

```
> image(t(next.gen),col=rainbow(100));v()
```



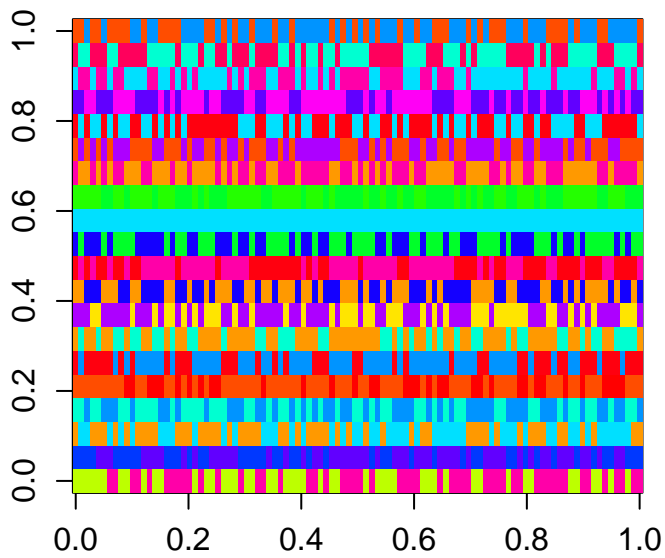
```
>
```

That's it! You can see how every line has input from two colors. Now we need to do this 300 times. Let us write it as a function.

```

> new.gen2=function(pop){
  mom.i=sample(1:20,rep=T); dad.i=sample(1:20,rep=T)
  mom=t( sapply( mom.i, function(i) pop[i,] ) )
  dad=t( sapply( dad.i, function(i) pop[i,] ) )
  rec=matrix(sample(c(T,F), 20*100,rep=T),20,100)
  next.gen=mom
  next.gen[rec]=dad[rec]
  next.gen
}
+++ + + + + + + + >
> image( t(new.gen2(pop)), col=rainbow(100) );v()

```



```

> unique(c(pop))
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
>

```

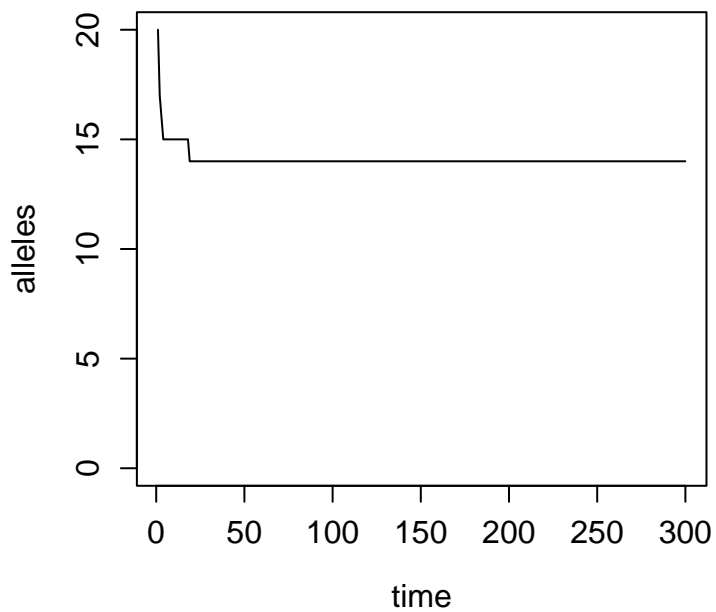
seems to work!

Now let us do it 300 times, but keep a record of alleles from how many individuals we still have around.

```

> alleles=rep(0,300)
> for( time in 1:300 ) {
  alleles[time] = length( unique( c( pop ) ) )
  pop = new.gen2(pop)
}
+++
>>
> plot(alleles,type="l",ylim=c(0,20),xlab="time");v()

```



>

We drop very quickly to alleles from 14 individuals left. Theoretically, we should see alleles from 75% of the population.

4. Read the data from the file `speeding.csv`. Calculate the mean speed. How many individuals were caught speeding before? Calculate the mean speed for those that were caught, and for those that were not.

```
> speeding=read.table("speeding.csv",head=T,sep=",",row.names=1)
```

```
> speeding[1:4,]
```

	speed	previous.ticket
1	124.68531	Yes
2	72.22792	No
3	95.54229	Yes
4	103.58920	No

```
> mean( speeding$speed )
```

```
[1] 107.8486
```

```
> summary(speeding)
```

	speed	previous.ticket
Min.	: 72.23	No :46
1st Qu.:	99.06	Yes:54
Median	:108.01	
Mean	:107.85	
3rd Qu.:	117.63	
Max.	:137.77	

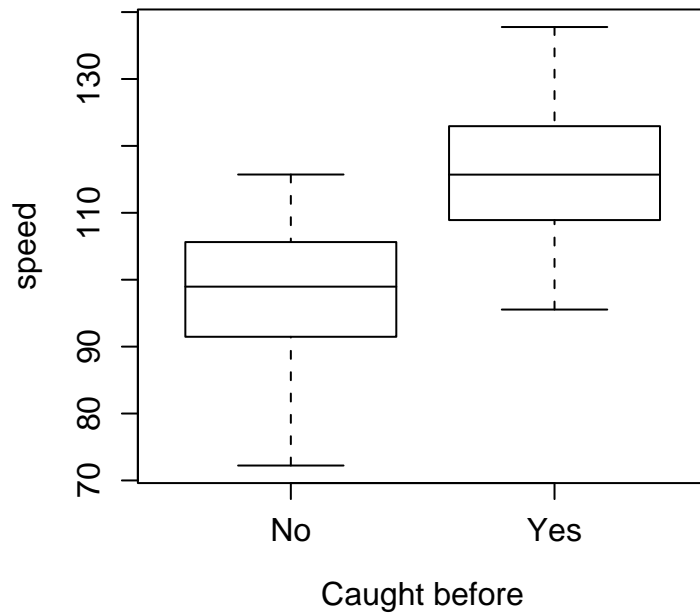
```
> mean( speeding$speed[ speeding$prev == "Yes" ] )
```

```
[1] 116.0982
```

```
> mean( speeding$speed[ speeding$prev == "No" ] )
```

```
[1] 98.16437
```

```
> plot(speeding$prev,speeding$speed,xlab="Caught before",ylab="speed");v()
```



```
>
```