1 Create allele distribution

First create a landscape. Here we use 10 populations with 2 stages per population. Each locus is going to start mononmorphic. To keep run times down, we will increase $4N_e\mu$ by increasing μ. Relatively few individuals are much quicker to simulate. Also we are setting the dispersal rates so high, the entire system should act like a single population

```
> library(rmetasim)
> seedMigrationRate <- 1
> pollenMigrationRate <- 1
> habitats <- 10
> stages <- 2
> carryperpop <- 100
> mu <- 0.001
> rland <- NULL
> rland <- new.landscape.empty()
> rland <- new.intparam.land(rland, h = habitats, s = stages, totgen = 10001)
> rland <- new.switchparam.land(rland, mp = 0)
> rland <- new.floatparam.land(rland)
> S <- matrix(c(0, 0, 1, 0), nrow = 2, byrow = TRUE)
> R <- matrix(c(0, 1.1, 0, 0), nrow = 2, byrow = TRUE)
> M <- matrix(c(0, 0, 0, 1), nrow = 2, byrow = TRUE)
> rland <- new.local.demo(rland, S, R, M)
> rland <- new.epoch.island(rland, 0, c(0, 0), c(0, 0), seedMigrationRate,
+     c(1, 0), c(1, 0), pollenMigrationRate, c(0, 1), c(0, 1),
+     carry = rep(carryperpop, habitats))
> rland <- new.locus(rland, type = 2, ploidy = 1, transmission = 1,
+     numalleles = 1, allelesize = 100, mutationrate = mu)
> for (x in 1:9) {
+     rland <- new.locus(rland, type = 2, ploidy = 2, transmission = 0,
+         numalleles = 1, allelesize = 100, mutationrate = mu)
+ }
> rland <- new.individuals(rland, c(500, 0, 500, 0, 500, 0, 500,
+     0, 500, 0, 500, 0, 500, 0, 500, 0, 500, 0, 500, 0))
```

1.1 Run simulation

Simulation runs for 1000 generations total. Every 20 generations, the state of the landscape is saved in three ways (as demonstration): to a r binary data file ("current.Rdata"), a metasim text file ("current.dat"), and as an element pushed onto an R list called l.exp (not actually written to disk). This example is certainly overkill in terms of formats, but it's probably a good idea to write the state of the landscape out periodically during a really long simulation.

This code sets up a simulation and runs it numsteps*stepsize generations and repeats from the same starting conditions numreps times. The state of all sampled landscapes are stored in the structure l.exp. (l.exp[[1]][[2]] is the first replicate, second time-click, etc..).

```
> numreps <- 1
> numsteps <- 5
> stepsize <- 200
> l.exp <- list(numreps)
> rland.start <- rland
> for (j in 1:numreps) {
+     rland <- rland.start
+     l.exp[[j]] <- list((2 * numsteps) + 1)
+     l.exp[[j]][[1]] <- rland
+     for (i in 1:numsteps) {
+         rland <- simulate.landscape(rland, stepsize)
+         save(rland, file = "current.Rdata")
+         write.landscape(rland, "current.dat")
+         l.exp[[j]][[i + 1]] <- rland
+     }
+ }
```

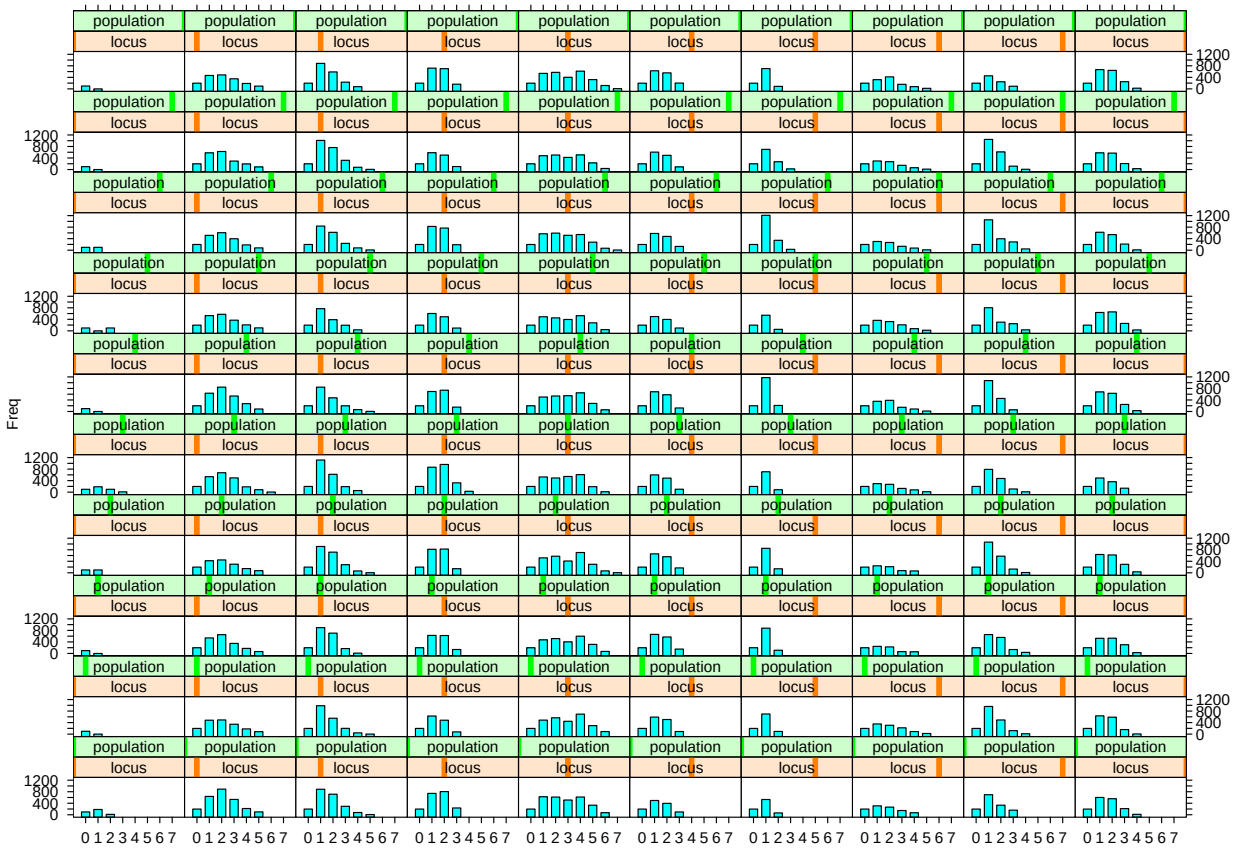
1.2 Plot mismatch distribution

This figure gives the mismatch dists at each locus. For lots of populations and loci, it is easier to read on X11 or Aqua than printed.

Write a graphics file with some special parameters (color, namely):

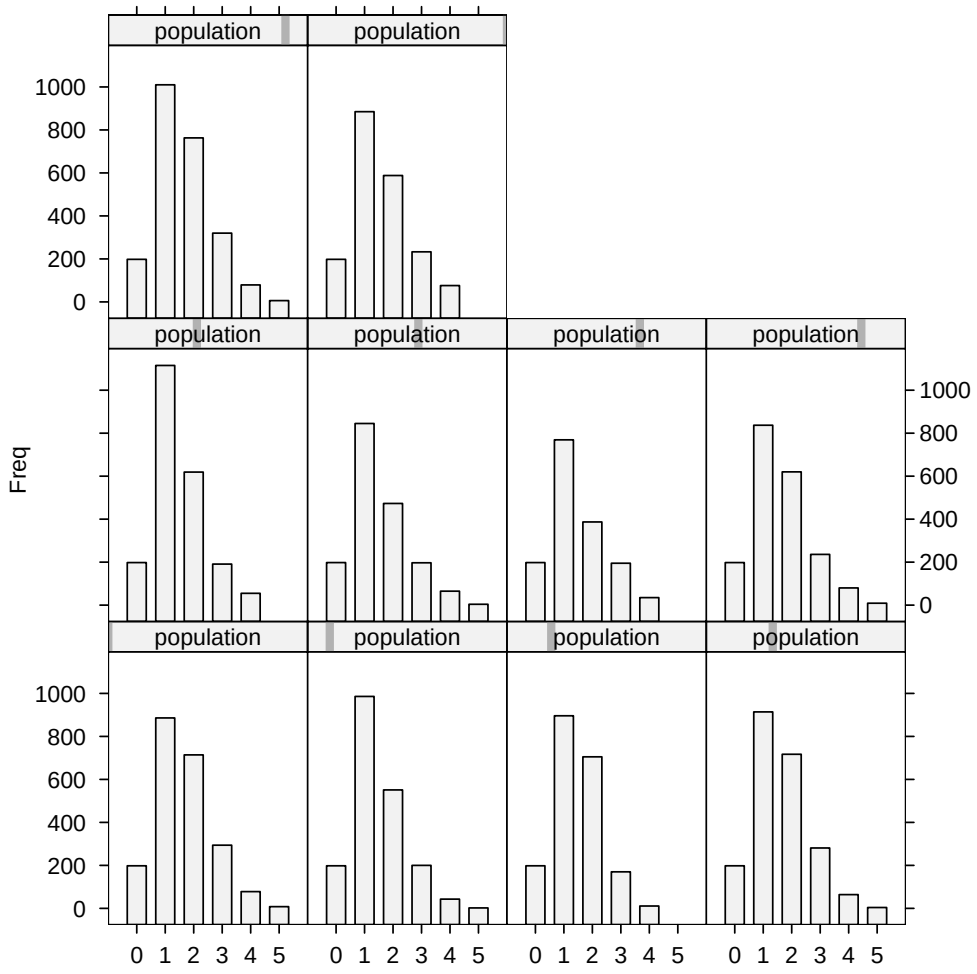
```
> mismatch.df <- mismatch.pop(rland)
> library(lattice)
> trellis.device("postscript", color = TRUE, file = "mismatch1.eps")
> print(barchart(Freq ~ ntdiff | locus * population, data = mismatch.df))
> dev.off()

null device
          1
```

Now include the graphic image:



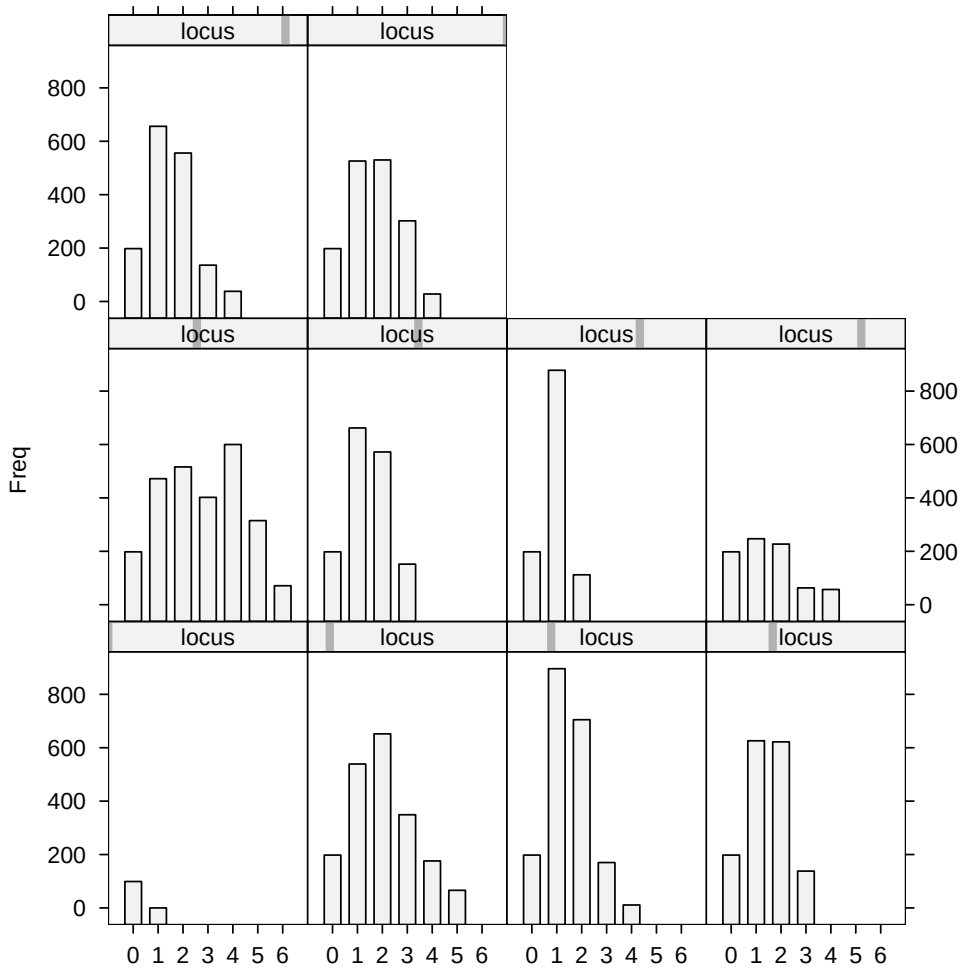
Here is a slice of one locus per population

```
> locnum <- 3
> mismatch.local <- mismatch.df[(mismatch.df$locus == locnum),
+     ]
> print(barchart(Freq ~ ntdiff | population, data = mismatch.local))
```



And here is an example of all loci in a single population

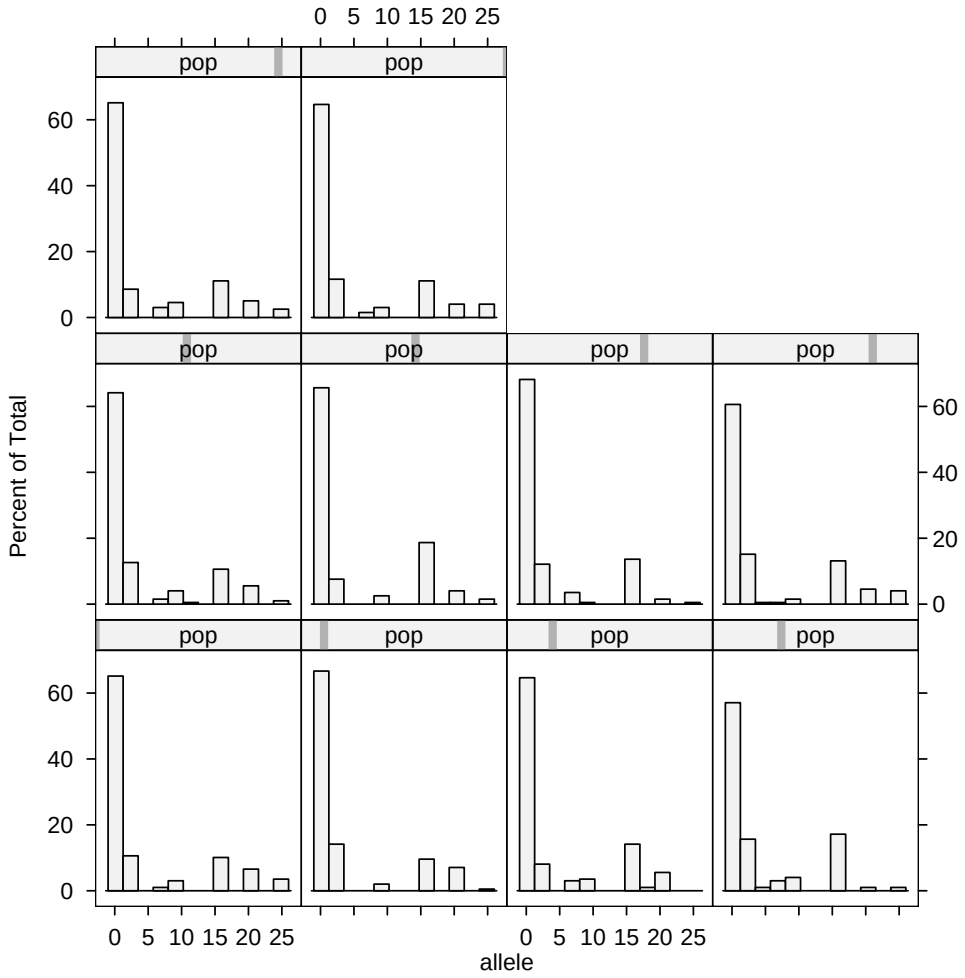
```
> pnum <- 3
> mismatch.local <- mismatch.df[(mismatch.df$population == pnum),
+     ]
> print(barchart(Freq ~ ntdiff | locus, data = mismatch.local))
```



1.3 Allele distributions

The previous (and some subsequent) section illustrates mismatch distributions. This figure gives the distribution of *alleles* in each population for a single locus. The full population * locus plot can be generated by histogram as well, but like the mismatch distribution, it does not reproduce well in printed format.

```
> ltr <- landscape.to.rtheta(rland)
> ltr.local <- ltr[ltr$locus == 3, ]
> print(histogram(allele | pop, data = ltr.local))
```



1.4 Modify the simulation part way through.

$\theta(= 4N_e\mu)$ for diploid loci is approximately 4 so far in this simulation (4 * 1000 individuals * 0.001).

Now we reduce migration between populations to a much lower number. The direct approach is to manipulate the epoch matrices "R" and "M". Also, make populations 1 and 4 twice as big and 6 and 10 half the current size

```
> epoch <- l.exp[[1]][[numsteps + 1]]$demography$epochs[[1]]
> print(epoch$R)
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
[1,]    0  1.1    1  0.0    1  0.0    1  0.0    1   0.0     1   0.0     1
[2,]    0  0.0    0  0.0    0  0.0    0  0.0    0   0.0     0   0.0     0
[3,]    1  0.0    0  1.1    1  0.0    1  0.0    1   0.0     1   0.0     1
[4,]    0  0.0    0  0.0    0  0.0    0  0.0    0   0.0     0   0.0     0
```

```
 [5,]    1  0.0    1  0.0    0  1.1    1  0.0    1   0.0     1  0.0     1
 [6,]    0  0.0    0  0.0    0  0.0    0  0.0    0   0.0     0  0.0     0
 [7,]    1  0.0    1  0.0    1  0.0    0  1.1    1   0.0     1  0.0     1
 [8,]    0  0.0    0  0.0    0  0.0    0  0.0    0   0.0     0  0.0     0
 [9,]    1  0.0    1  0.0    1  0.0    1  0.0    0   1.1     1  0.0     1
[10,]    0  0.0    0  0.0    0  0.0    0  0.0    0   0.0     0  0.0     0
[11,]    1  0.0    1  0.0    1  0.0    1  0.0    1   0.0     0  1.1     1
[12,]    0  0.0    0  0.0    0  0.0    0  0.0    0   0.0     0  0.0     0
[13,]    1  0.0    1  0.0    1  0.0    1  0.0    1   0.0     1  0.0     0
[14,]    0  0.0    0  0.0    0  0.0    0  0.0    0   0.0     0  0.0     0
[15,]    1  0.0    1  0.0    1  0.0    1  0.0    1   0.0     1  0.0     1
[16,]    0  0.0    0  0.0    0  0.0    0  0.0    0   0.0     0  0.0     0
[17,]    1  0.0    1  0.0    1  0.0    1  0.0    1   0.0     1  0.0     1
[18,]    0  0.0    0  0.0    0  0.0    0  0.0    0   0.0     0  0.0     0
[19,]    1  0.0    1  0.0    1  0.0    1  0.0    1   0.0     1  0.0     1
[20,]    0  0.0    0  0.0    0  0.0    0  0.0    0   0.0     0  0.0     0
      [,14] [,15] [,16] [,17] [,18] [,19] [,20]
 [1,]   0.0     1   0.0     1   0.0     1   0.0
 [2,]   0.0     0   0.0     0   0.0     0   0.0
 [3,]   0.0     1   0.0     1   0.0     1   0.0
 [4,]   0.0     0   0.0     0   0.0     0   0.0
 [5,]   0.0     1   0.0     1   0.0     1   0.0
 [6,]   0.0     0   0.0     0   0.0     0   0.0
 [7,]   0.0     1   0.0     1   0.0     1   0.0
 [8,]   0.0     0   0.0     0   0.0     0   0.0
 [9,]   0.0     1   0.0     1   0.0     1   0.0
[10,]   0.0     0   0.0     0   0.0     0   0.0
[11,]   0.0     1   0.0     1   0.0     1   0.0
[12,]   0.0     0   0.0     0   0.0     0   0.0
[13,]   1.1     1   0.0     1   0.0     1   0.0
[14,]   0.0     0   0.0     0   0.0     0   0.0
[15,]   0.0     0   1.1     1   0.0     1   0.0
[16,]   0.0     0   0.0     0   0.0     0   0.0
[17,]   0.0     1   0.0     0   1.1     1   0.0
[18,]   0.0     0   0.0     0   0.0     0   0.0
[19,]   0.0     1   0.0     1   0.0     0   1.1
[20,]   0.0     0   0.0     0   0.0     0   0.0
```

> print(epoch$M)

```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
 [1,]    0    0    0    0    0    0    0    0    0     0     0     0     0
 [2,]    0    1    0    1    0    1    0    1    0     1     0     1     0
```

```
 [3,]    0    0    0    0    0    0    0    0    0    0    0    0    0
 [4,]    0    1    0    1    0    1    0    1    0    1    0    1    0
 [5,]    0    0    0    0    0    0    0    0    0    0    0    0    0
 [6,]    0    1    0    1    0    1    0    1    0    1    0    1    0
 [7,]    0    0    0    0    0    0    0    0    0    0    0    0    0
 [8,]    0    1    0    1    0    1    0    1    0    1    0    1    0
 [9,]    0    0    0    0    0    0    0    0    0    0    0    0    0
[10,]    0    1    0    1    0    1    0    1    0    1    0    1    0
[11,]    0    0    0    0    0    0    0    0    0    0    0    0    0
[12,]    0    1    0    1    0    1    0    1    0    1    0    1    0
[13,]    0    0    0    0    0    0    0    0    0    0    0    0    0
[14,]    0    1    0    1    0    1    0    1    0    1    0    1    0
[15,]    0    0    0    0    0    0    0    0    0    0    0    0    0
[16,]    0    1    0    1    0    1    0    1    0    1    0    1    0
[17,]    0    0    0    0    0    0    0    0    0    0    0    0    0
[18,]    0    1    0    1    0    1    0    1    0    1    0    1    0
[19,]    0    0    0    0    0    0    0    0    0    0    0    0    0
[20,]    0    1    0    1    0    1    0    1    0    1    0    1    0
      [,14] [,15] [,16] [,17] [,18] [,19] [,20]
 [1,]     0     0     0     0     0     0     0
 [2,]     1     0     1     0     1     0     1
 [3,]     0     0     0     0     0     0     0
 [4,]     1     0     1     0     1     0     1
 [5,]     0     0     0     0     0     0     0
 [6,]     1     0     1     0     1     0     1
 [7,]     0     0     0     0     0     0     0
 [8,]     1     0     1     0     1     0     1
 [9,]     0     0     0     0     0     0     0
[10,]     1     0     1     0     1     0     1
[11,]     0     0     0     0     0     0     0
[12,]     1     0     1     0     1     0     1
[13,]     0     0     0     0     0     0     0
[14,]     1     0     1     0     1     0     1
[15,]     0     0     0     0     0     0     0
[16,]     1     0     1     0     1     0     1
[17,]     0     0     0     0     0     0     0
[18,]     1     0     1     0     1     0     1
[19,]     0     0     0     0     0     0     0
[20,]     1     0     1     0     1     0     1
```

```
> epoch$R <- epoch$R * 0.01
> epoch$M <- epoch$M * 0.01
> epoch$Carry <- epoch$Carry * c(2, 1, 1, 2, 1, 0.5, 1, 1, 1, 0.5)
```

```
> l.exp[[1]][[numsteps + 1]]$demography$epochs[[1]] <- epoch
> print(l.exp[[1]][[numsteps + 1]]$demography$epochs[[1]]$R)

      [,1]  [,2] [,3]  [,4] [,5]  [,6] [,7]  [,8] [,9] [,10] [,11] [,12] [,13]
 [1,] 0.00 0.011 0.01 0.000 0.01 0.000 0.01 0.000 0.01 0.000  0.01 0.000  0.01
 [2,] 0.00 0.000 0.00 0.000 0.00 0.000 0.00 0.000 0.00 0.000  0.00 0.000  0.00
 [3,] 0.01 0.000 0.00 0.011 0.01 0.000 0.01 0.000 0.01 0.000  0.01 0.000  0.01
 [4,] 0.00 0.000 0.00 0.000 0.00 0.000 0.00 0.000 0.00 0.000  0.00 0.000  0.00
 [5,] 0.01 0.000 0.01 0.000 0.00 0.011 0.01 0.000 0.01 0.000  0.01 0.000  0.01
 [6,] 0.00 0.000 0.00 0.000 0.00 0.000 0.00 0.000 0.00 0.000  0.00 0.000  0.00
 [7,] 0.01 0.000 0.01 0.000 0.01 0.000 0.00 0.011 0.01 0.000  0.01 0.000  0.01
 [8,] 0.00 0.000 0.00 0.000 0.00 0.000 0.00 0.000 0.00 0.000  0.00 0.000  0.00
 [9,] 0.01 0.000 0.01 0.000 0.01 0.000 0.01 0.000 0.00 0.011  0.01 0.000  0.01
[10,] 0.00 0.000 0.00 0.000 0.00 0.000 0.00 0.000 0.00 0.000  0.00 0.000  0.00
[11,] 0.01 0.000 0.01 0.000 0.01 0.000 0.01 0.000 0.01 0.000  0.00 0.011  0.01
[12,] 0.00 0.000 0.00 0.000 0.00 0.000 0.00 0.000 0.00 0.000  0.00 0.000  0.00
[13,] 0.01 0.000 0.01 0.000 0.01 0.000 0.01 0.000 0.01 0.000  0.01 0.000  0.00
[14,] 0.00 0.000 0.00 0.000 0.00 0.000 0.00 0.000 0.00 0.000  0.00 0.000  0.00
[15,] 0.01 0.000 0.01 0.000 0.01 0.000 0.01 0.000 0.01 0.000  0.01 0.000  0.01
[16,] 0.00 0.000 0.00 0.000 0.00 0.000 0.00 0.000 0.00 0.000  0.00 0.000  0.00
[17,] 0.01 0.000 0.01 0.000 0.01 0.000 0.01 0.000 0.01 0.000  0.01 0.000  0.01
[18,] 0.00 0.000 0.00 0.000 0.00 0.000 0.00 0.000 0.00 0.000  0.00 0.000  0.00
[19,] 0.01 0.000 0.01 0.000 0.01 0.000 0.01 0.000 0.01 0.000  0.01 0.000  0.01
[20,] 0.00 0.000 0.00 0.000 0.00 0.000 0.00 0.000 0.00 0.000  0.00 0.000  0.00
      [,14] [,15] [,16] [,17] [,18] [,19] [,20]
 [1,] 0.000  0.01 0.000  0.01 0.000  0.01 0.000
 [2,] 0.000  0.00 0.000  0.00 0.000  0.00 0.000
 [3,] 0.000  0.01 0.000  0.01 0.000  0.01 0.000
 [4,] 0.000  0.00 0.000  0.00 0.000  0.00 0.000
 [5,] 0.000  0.01 0.000  0.01 0.000  0.01 0.000
 [6,] 0.000  0.00 0.000  0.00 0.000  0.00 0.000
 [7,] 0.000  0.01 0.000  0.01 0.000  0.01 0.000
 [8,] 0.000  0.00 0.000  0.00 0.000  0.00 0.000
 [9,] 0.000  0.01 0.000  0.01 0.000  0.01 0.000
[10,] 0.000  0.00 0.000  0.00 0.000  0.00 0.000
[11,] 0.000  0.01 0.000  0.01 0.000  0.01 0.000
[12,] 0.000  0.00 0.000  0.00 0.000  0.00 0.000
[13,] 0.011  0.01 0.000  0.01 0.000  0.01 0.000
[14,] 0.000  0.00 0.000  0.00 0.000  0.00 0.000
[15,] 0.000  0.00 0.011  0.01 0.000  0.01 0.000
[16,] 0.000  0.00 0.000  0.00 0.000  0.00 0.000
[17,] 0.000  0.01 0.000  0.00 0.011  0.01 0.000
[18,] 0.000  0.00 0.000  0.00 0.000  0.00 0.000
```

```
[19,] 0.000  0.01 0.000  0.01 0.000  0.00 0.011
[20,] 0.000  0.00 0.000  0.00 0.000  0.00 0.000
```

1.5 Run second portion of the simulation

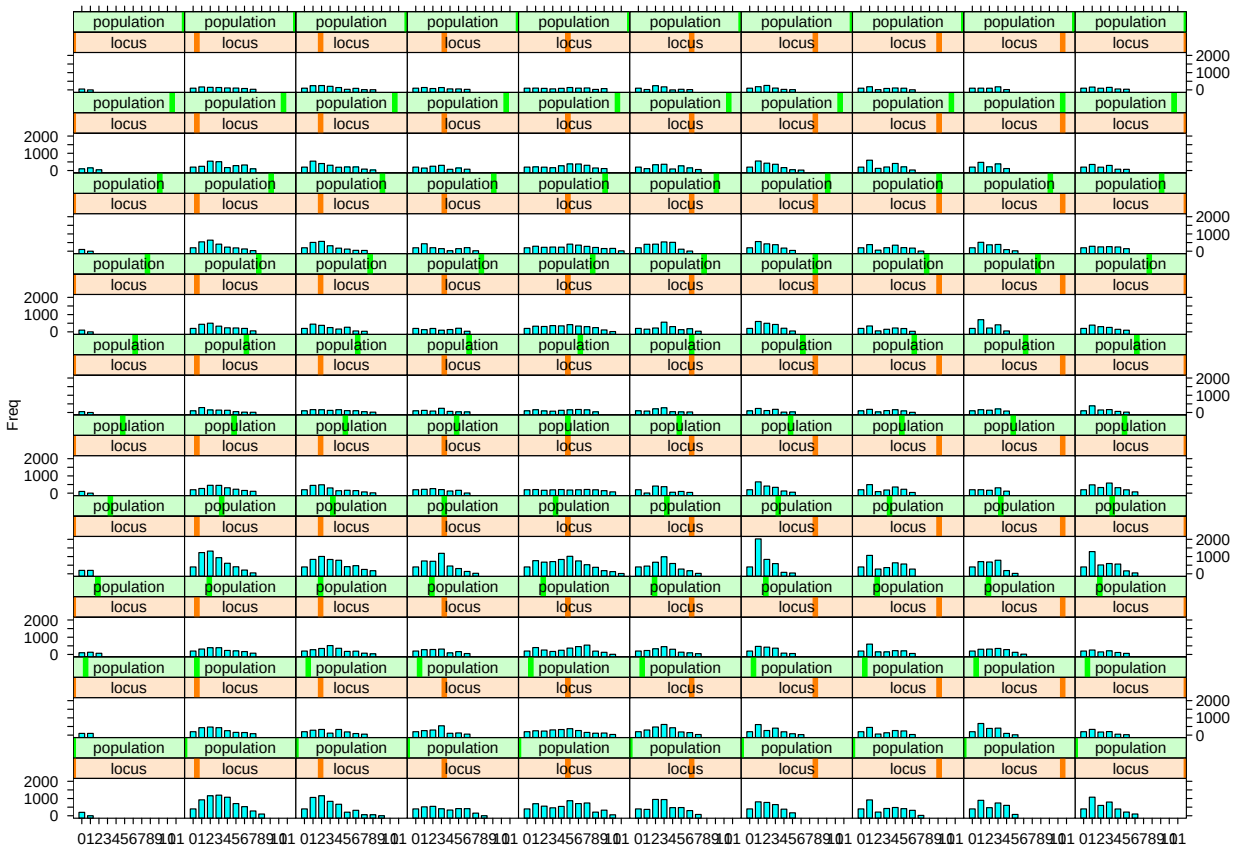
```
> for (j in 1:numreps) {
+     rland <- l.exp[[1]][[numsteps + 1]]
+     for (i in (numsteps + 1):(2 * numsteps) + 1) {
+         rland <- simulate.landscape(rland, stepsize)
+         save(rland, file = "current.Rdata")
+         write.landscape(rland, "current.dat")
+         l.exp[[j]][[i]] <- rland
+     }
+ }
```

1.6 Look at the mismatch dists at the end of the second sim

Write a graphics file with some special parameters (color, namely):

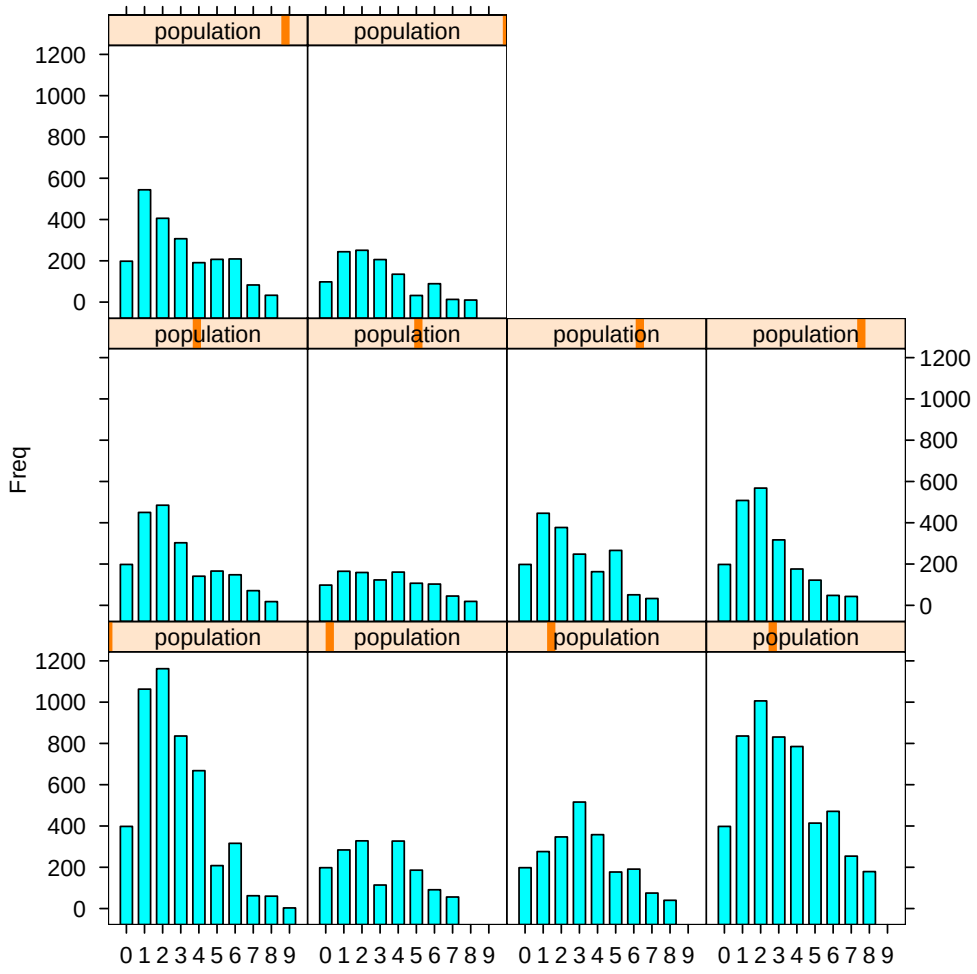
```
> mismatch.df <- mismatch.pop(rland)
> library(lattice)
> trellis.device("postscript", color = TRUE, file = "mismatch2.eps")
> print(barchart(Freq ~ ntdiff | locus * population, data = mismatch.df))
> dev.off()

postscript
         2
```

Now include the graphic image:



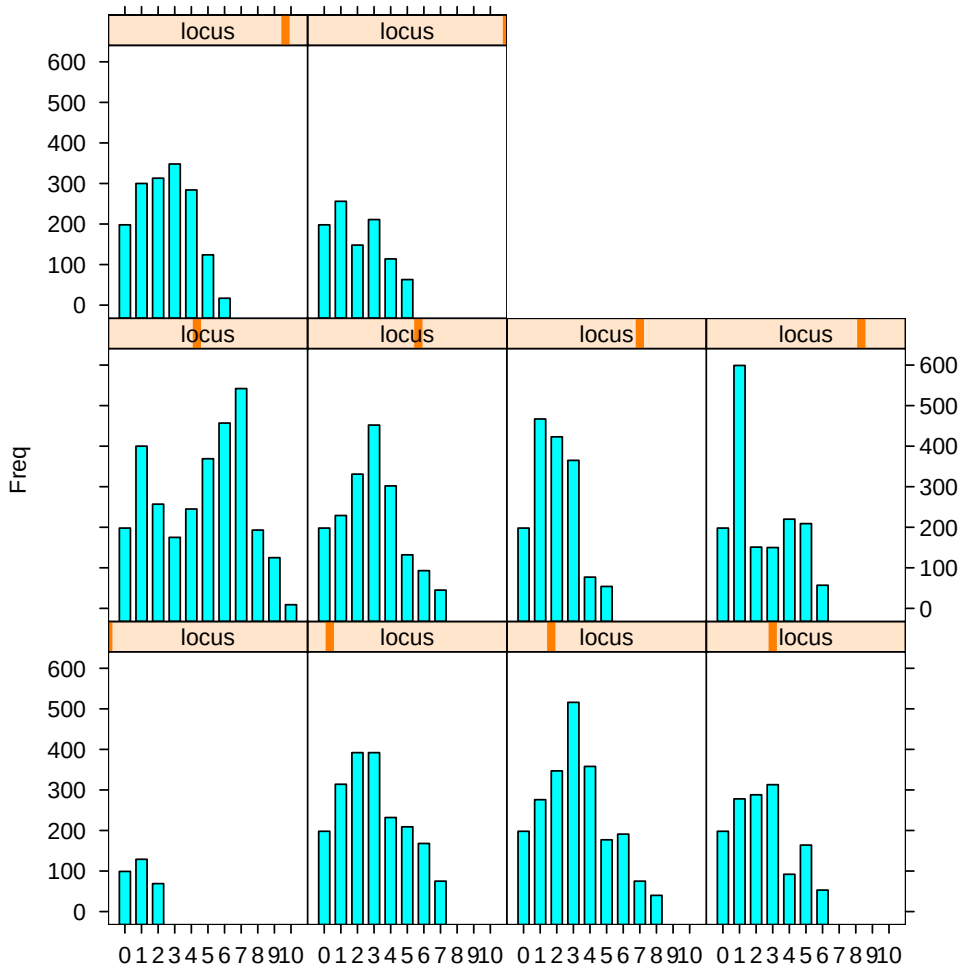
Here is a slice of one locus per population.

```
> locnum <- 3
> mismatch.local <- mismatch.df[(mismatch.df$locus == locnum),
+     ]
> print(barchart(Freq ~ ntdiff | population, data = mismatch.local))
```



And here is an example of all loci in a single population

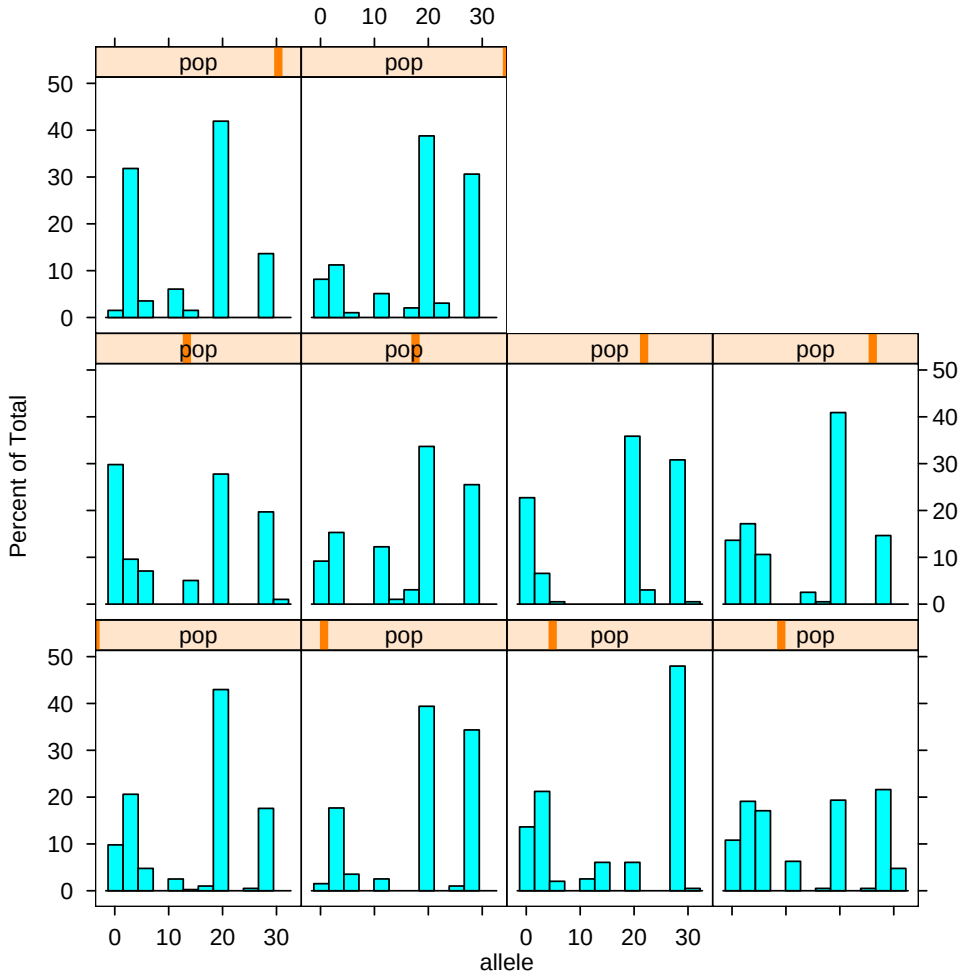
```
> pnum <- 3
> mismatch.local <- mismatch.df[(mismatch.df$population == pnum),
+     ]
> print(barchart(Freq ~ ntdiff | locus, data = mismatch.local))
```



1.7 Allele distributions

This figure gives the distribution of *alleles* in each population at the end of the simulation.

```
> ltr <- landscape.to.rtheta(rland)
> ltr.local <- ltr[ltr$locus == 3, ]
> print(histogram(allele | pop, data = ltr.local))
```



Last line of document: generated with Sweave