landgensim_tutorial

Bernd Gruber

22 February 2016

LandgenSimulation: A tutorial how to simulate a metapopulation on a resistance landscape using landgenreport function

Setup your computer

Install packages

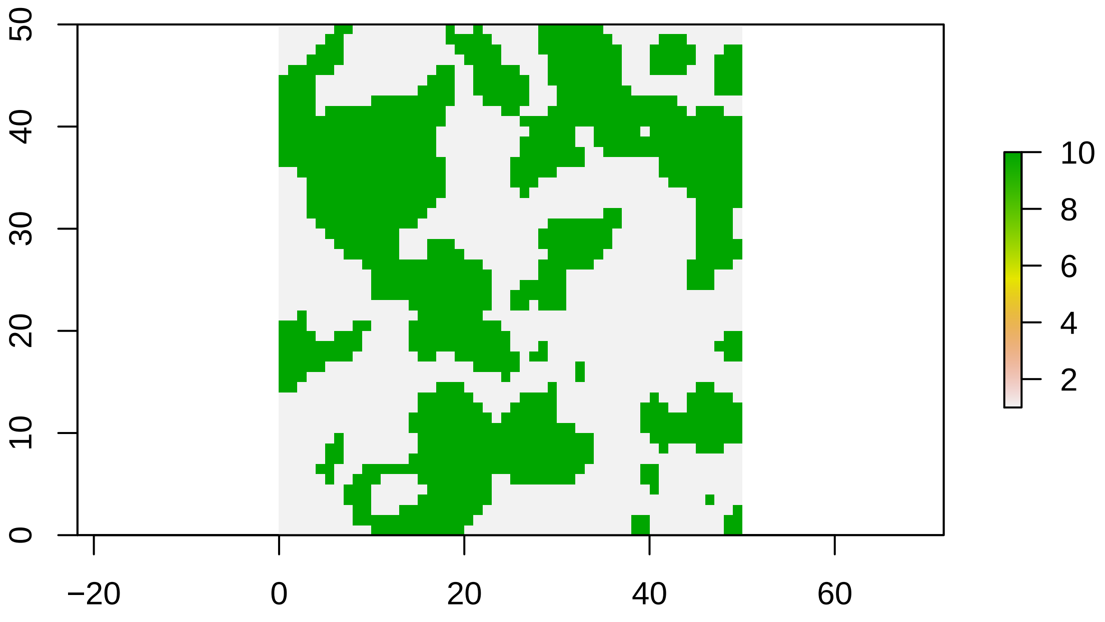
We need to install the latest PopGenReport package (see www.popgenreport.org) and load this package and some additional packages

```r
library(PopGenReport )  #load the package
library(secr)  #to create a random habitat
library(gdistance)
```

Create a random landscape

We will use the randomHabitat() function from the secr package, but you can use mutiple different ways. E.g. simply load a png file or any other file format using the raster function (?raster::raster, see the examples in there).

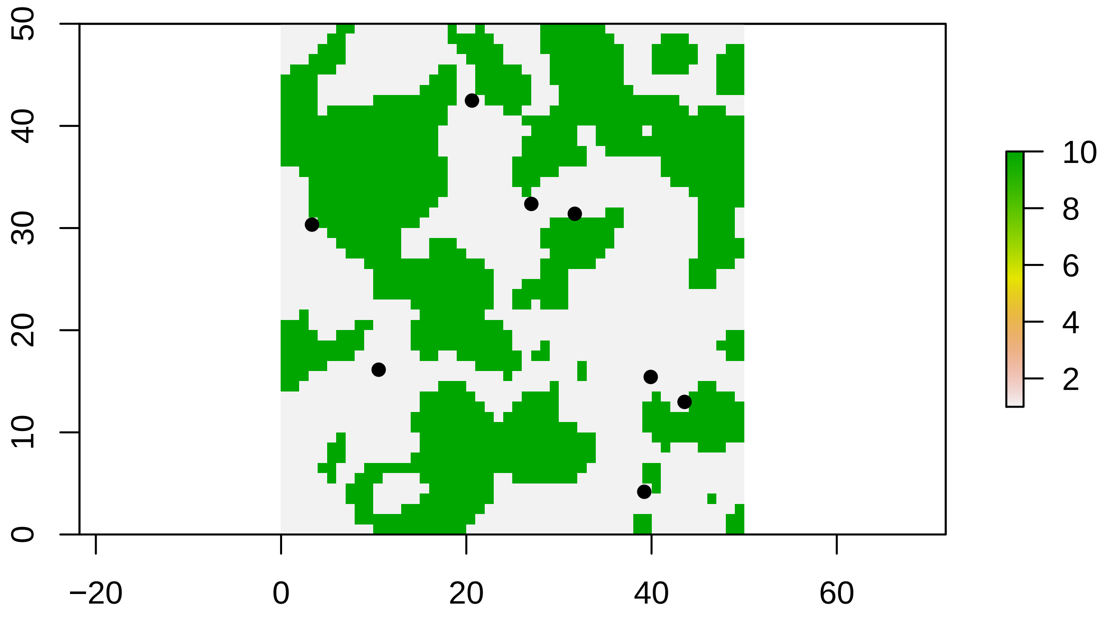
```r
nx=50
ny=50
set.seed(555) #(to make sure we have the same example running)
tempmask<-make.mask(nx=nx,ny=ny,spacing=1)
r <- raster(randomHabitat(tempmask, p = 0.5, A = 0.5))
#set non-habitat to friction values of 10
values(r)[is.na(values(r))==T]<-10
plot(r)
```



Add populations to the landscape (using minimal distance)

```
#we create a function that allows us to set up n subpopulations in the habitat only (non green areas)
#the subpopulations should be at least mindist units apart
createpops <- function(n, mindist, landscape, plot=TRUE)
{

minx <- extent(landscape)@xmin #get the min and max  coordinates
miny <- extent(landscape)@ymin #coordinates of the landscape
maxx <- extent(landscape)@xmax
maxy <- extent(landscape)@ymax

cc<- 1
coords <- data.frame(lx=NA, ly=NA)
while (cc<= n )  #repeat until you have found n locations
{
  draw=FALSE
  while (draw==FALSE)
  {
    x <- runif(1,minx,maxx)
    y <- runif(1,miny,maxy)
    if (landscape[cellFromXY(landscape,c(x,y) )]==1)  draw=TRUE #check if in the habitat
  }
```

```r
coords[cc,] <- c(x,y)

if (nrow(coords)>1) d <- min(dist(coords)) else d <- mindist+1

if (d > mindist) cc <- cc+1  #take the location only if distance is larger than mindist
}
if (plot==TRUE)
{
plot(landscape)
points(coords, pch=16)
}
return( as.matrix( coords))
}

#test the function above.....
createpops(n=8, mindist = 3, landscape = r, plot = TRUE)
```



```
##           lx        ly
## 1 39.909277 15.423014
## 2 20.610496 42.481624
## 3 39.214161  4.181753
## 4  3.336042 30.336679
## 5 27.020272 32.362852
## 6 31.711251 31.395343
## 7 43.562872 12.980376
## 8 10.537505 16.133367
```

Initialise a metapopulation

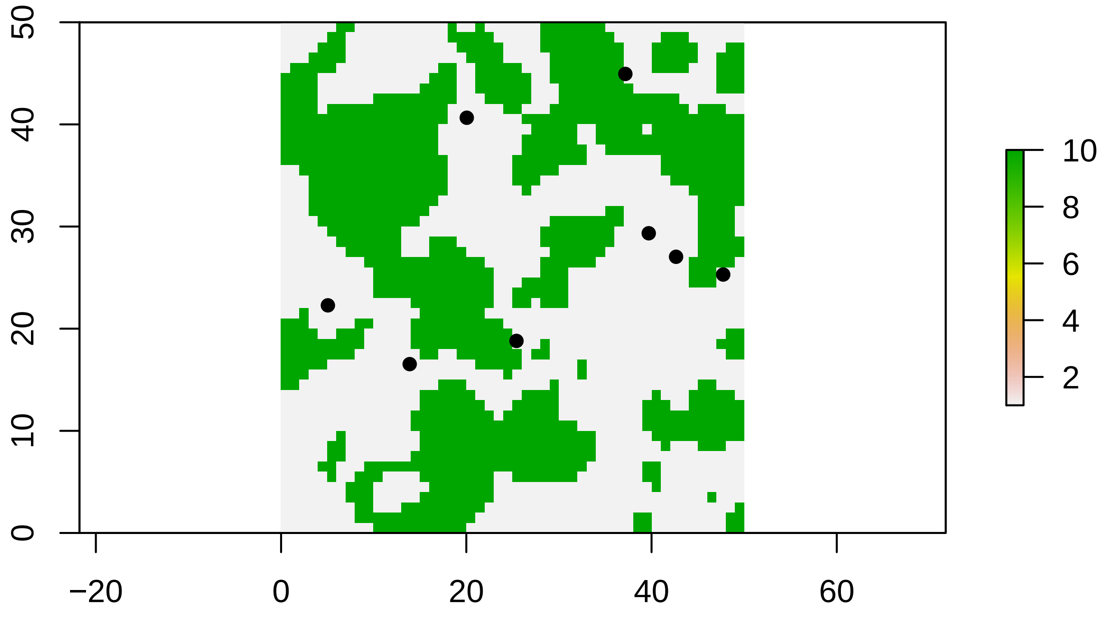
We use init.popgensim to initialise a metapopulation based on the locations we created earlier. To do this we need to initialise a number of parameters (the locations of the subpopulations, the number of individuals per subpopulation, the number of loci and alleles per loci. For a full list check ?init.popgensim)

To store all the parameters we create a list called para where we store all of them

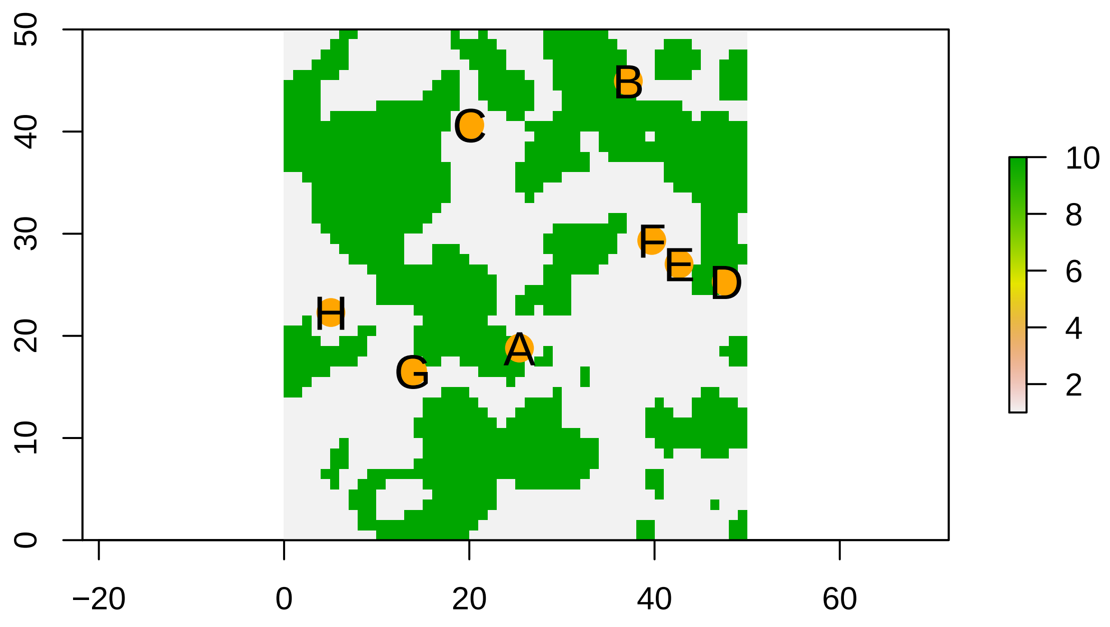
A) Define your metapopulation

```
################################################################################
#### Define Metapopulation
################################################################################

para<- list()
#Define populations (dynamics)
para$n.pops=8
para$n.ind=100

para$sex.ratio <- 0.5
#age distribution....

para$n.cov <- 3 #number of covariates (before the loci in the data.frame, do not change this!!)


################################################################################
#### Define Popdynamics
################################################################################

#reproduction
para$n.offspring = 2

#migration
para$mig.rate <- 0.1

#dispersal: exponential dispersal with maximal distance in map units
para$disp.max=50   #average  dispersal of an individual in meters
para$disp.rate = 0.05 #proportion of dispersing individuals

#Define genetics
para$n.allels <- 10
para$n.loci <- 20
para$mut.rate <- 0.001

################################################################################
#### Define cost distance method
################################################################################
```

```
para$method <- "leastcost" #rSPDdistance, commute
para$NN <- 8  #number of neighbours for the cost distance method

# A)    init simulation populations from scratch
landscape<- r  #<-raster(system.file("external/rlogo.grd", package="raster"))


 #Define x and y locations
 para$locs <-createpops(n=para$n.pops, mindist = 3, landscape = r, plot = TRUE)
```



```
 #give the population some names
 rownames(para$locs) <- LETTERS[1:para$n.pops]


  #create a costdistance matrix
  cost.mat <- costdistances(landscape, para$locs, para$method, para$NN) #needed for the simulation
  eucl.mat <- as.matrix(dist(para$locs))  #needed for the analysis later


#Plot your landscape with the populations....

  plot(landscape)
  points(para$locs[,1], para$locs[,2], pch=16, cex=2, col="orange")
  text(para$locs[,1],para$locs[,2], row.names(para$locs), cex=1.5)
```



```r
#check the parameter list

  para
```

```
## $n.pops
## [1] 8
##
## $n.ind
## [1] 100
##
## $sex.ratio
## [1] 0.5
##
## $n.cov
## [1] 3
##
## $n.offspring
## [1] 2
##
## $mig.rate
## [1] 0.1
##
## $disp.max
## [1] 50
##
## $disp.rate
## [1] 0.05
##
## $n.allels
## [1] 10
##
## $n.loci
## [1] 20
##
## $mut.rate
## [1] 0.001
##
## $method
## [1] "leastcost"
##
## $NN
## [1] 8
##
## $locs
##          lx       ly
## A 25.414122 18.80199
## B 37.170582 44.94059
## C 20.049062 40.65154
## D 47.735499 25.29914
## E 42.647544 27.03850
## F 39.695677 29.34214
## G 13.882835 16.53103
## H  5.052738 22.28383
```

B) initialise your population on the landscape

Now finally we can initialise our population using the init function

```
# B) initialise your population on the landscape
  simpops <- init.popgensim(para$n.pops, para$n.ind, para$sex.ratio, para$n.loci, para$n.allels, para$lo
```

You may want to check the simpops object, which is simply a list of our subpopulation and each individual is coded in a single run in one of the subpopulations.

```
names(simpops)  #the names of the subpopulations
```

```
## [1] "A" "B" "C" "D" "E" "F" "G" "H"
```

```
head(simpops$A) # a list of the first 6 individuals of population A
```

```
##   pop    sex age locus1A locus1B locus2A locus2B locus3A locus3B locus4A
## 1   1 female  NA       8      10       9       8       9       2       2
## 2   1 female  NA       3       5       3       3       1       9       4
## 3   1 female  NA       6       1       5       7      10       6       2
## 4   1 female  NA       7       5       6       3       6       5       7
## 5   1 female  NA       8       3       5       1       6       7       7
## 6   1 female  NA       5       4       9       1       2       5       7
##   locus4B locus5A locus5B locus6A locus6B locus7A locus7B locus8A locus8B
## 1       2       4       2       3       1       8       5       7       5
## 2      10       8       1       9       6       5       6       1       8
## 3       9       8       9       2       4       1       6       6       9
## 4       8       7       2       4       8       8       6       1       6
## 5       5       1       8       1       4       3       6       1       5
## 6       7       7       2      10       3       4       8       1       8
##   locus9A locus9B locus10A locus10B locus11A locus11B locus12A locus12B
## 1       8       1        5        8        2        2        7        6
## 2      10      10        3        1        1        1        4       10
## 3       2       9       10        6        6        6        3       10
## 4       9       4       10        4        5        1        8        5
## 5       6       7        9        8        7        7        7        9
## 6      10       3        7        1        2        7        4        5
##   locus13A locus13B locus14A locus14B locus15A locus15B locus16A locus16B
## 1        9        8        1        6        9        9        3        3
## 2       10       10        7        1        1        3        4       10
## 3        9        6        9        9       10        8        7        9
## 4        9        8        2        5        6        1        1        8
## 5        9       10        3        7        3        6        5        2
## 6        9        2        5        2        2       10        2        3
##   locus17A locus17B locus18A locus18B locus19A locus19B locus20A locus20B
## 1        7        6       10        4        4        9        5        7
## 2        5        4        6        8       10        6        5        3
## 3        1        2        5        3        2        6        3        5
## 4        8        3        2        8        8        7       10        2
## 5       10        8        3        8        5        7        2        7
## 6        6       10        2        3        5        2        8        6
```

We can also analyse our simpop object. (e.g. calculate the pairwise Fst value between all the populations.) To be able to do that we first need to convert it into a genind object (because many function need this type of object as input)

```
gsp <- pops2genind(simpops, locs =para$locs)

gsp #chekc the genind object
```

```
## /// GENIND OBJECT /////////
##
##  // 800 individuals; 20 loci; 200 alleles; size: 728 Kb
##
##  // Basic content
##    @tab:  800 x 200 matrix of allele counts
##    @loc.n.all: number of alleles per locus (range: 10-10)
##    @loc.fac: locus factor for the 200 columns of @tab
##    @all.names: list of allele names for each locus
##    @ploidy: ploidy of each individual  (range: 2-2)
##    @type:  codom
##    @call: df2genind(X = res, sep = "/", ind.names = rownames(res), pop = combine$pop)
##
##  // Optional content
##    @pop: population of each individual (group size range: 100-100)
##    @other: a list containing: xy
```

```r
summary(gsp)  #some summary statistics
```

```
##
##  # Total number of genotypes:  800
##
##  # Population sample sizes:
##   A   B   C   D   E   F   G   H
## 100 100 100 100 100 100 100 100
##
##  # Number of alleles per locus:
##  X1  X2  X3  X4  X5  X6  X7  X8  X9 X10 X11 X12 X13 X14 X15 X16 X17 X18
##  10  10  10  10  10  10  10  10  10  10  10  10  10  10  10  10  10  10
## X19 X20
##  10  10
##
##  # Number of alleles per population:
##   A   B   C   D   E   F   G   H
## 200 200 200 200 200 200 200 200
##
##  # Percentage of missing data:
## [1] 0
##
##  # Observed heterozygosity:
##      X1      X2      X3      X4      X5      X6      X7      X8      X9
## 0.91250 0.90250 0.91250 0.89625 0.89125 0.90750 0.87125 0.91250 0.89750
##     X10     X11     X12     X13     X14     X15     X16     X17     X18
## 0.89625 0.89750 0.92000 0.89125 0.89000 0.89000 0.90000 0.88000 0.89750
##     X19     X20
## 0.90000 0.90875
##
##  # Expected heterozygosity:
##        X1        X2        X3        X4        X5        X6        X7
## 0.8996977 0.8990570 0.8988820 0.8988984 0.8987875 0.8996633 0.8991445
##        X8        X9       X10       X11       X12       X13       X14
## 0.8997758 0.8993961 0.8997891 0.8995773 0.8992562 0.8996969 0.8993742
##       X15       X16       X17       X18       X19       X20
## 0.8994102 0.8995102 0.8997289 0.8989555 0.8992039 0.8995336
```

```r
library(mmod)

round(pairwise_Gst_Nei(gsp),3)
```

```
##        A      B      C      D      E      F      G
## B -0.001
## C  0.002  0.008
## D  0.002  0.005  0.000
## E  0.015  0.001 -0.002  0.001
## F  0.002  0.004 -0.003 -0.003  0.005
## G -0.002 -0.009 -0.001 -0.007 -0.001  0.001
## H -0.002 -0.004  0.002 -0.002  0.007 -0.002 -0.005
```

```
#is there an effect of the landscape on the population structure
#(there should not be after initialisation)
gen.mat <- pairwise.fstb(gsp) #around 150 times faster than mmod::pairwise_Gst_Nei, but slightly
#different....
 round(gen.mat ,3)
```

```
##       A     B     C     D     E     F     G     H
## A 0.000 0.002 0.003 0.003 0.003 0.003 0.002 0.002
## B 0.002 0.000 0.003 0.003 0.003 0.003 0.002 0.002
## C 0.003 0.003 0.000 0.003 0.002 0.002 0.002 0.003
## D 0.003 0.003 0.003 0.000 0.003 0.002 0.002 0.002
## E 0.003 0.003 0.002 0.003 0.000 0.003 0.002 0.003
## F 0.003 0.003 0.002 0.002 0.003 0.000 0.003 0.002
## G 0.002 0.002 0.002 0.002 0.002 0.003 0.000 0.002
## H 0.002 0.002 0.003 0.002 0.003 0.002 0.002 0.000
```

```
  # partial mantel test ?wassermann
  wassermann(eucl.mat = eucl.mat, cost.mats = list(cost=cost.mat), gen.mat = gen.mat,
             plot=F)$mantel.tab
```

```
##                    model       r     p
## 1 Gen ~cost | Euclidean  0.0371 0.402
## 2 Gen ~Euclidean | cost -0.2438 0.862
```

Check the pairwise Fst values, why are they so low?

Now we can run our simulation by simply passing our simpops, with some additional parameters that are needed for the simulation. The number of generation the simulation should run is in the steps parameter. (check ?run.popgensim for a description of all parameters).

Important to understand is the idea of the cost.mat (which is the cost matrix that is used for the distance between subpopulation). The n.alleles, n.ind cannot be different from the initialisation.

C) run your population years steps on the landscape

```
# C) run your population years steps on the landscape


  simpops <- run.popgensim(simpops, steps=3, cost.mat, n.offspring=para$n.offspring, n.ind=para$n.ind,
                           para$mig.rate, para$disp.max, para$disp.rate, para$n.allels, para$mut.rate,
                           n.cov=para$n.cov, rec="none")
```

In essence we were running a metapopulation with 100 individuals per subpopulation on our resistance landscape for 3 generations. The question is now was that enough time to create an effect on population structure? We should check now the pairwise Fst values and then do a landscape genetic analysis using partial mantel tests.

D) Analyse your simulated population using an LGA (partial mantel test)

```
#convert to genind to calculate pairwise fsts (this )
 gsp <- pops2genind(simpops, para$locs, para$n.cov)

 #calculate your genetic distance matrix e.g. fst or D
 gen.mat <- pairwise.fstb(gsp)
 round(gen.mat ,3)
```

```
##       A     B     C     D     E     F     G     H
## A 0.000 0.009 0.007 0.007 0.007 0.007 0.006 0.008
## B 0.009 0.000 0.008 0.008 0.006 0.007 0.007 0.008
## C 0.007 0.008 0.000 0.006 0.006 0.008 0.007 0.009
## D 0.007 0.008 0.006 0.000 0.007 0.006 0.006 0.007
## E 0.007 0.006 0.006 0.007 0.000 0.006 0.007 0.008
## F 0.007 0.007 0.008 0.006 0.006 0.000 0.007 0.008
## G 0.006 0.007 0.007 0.006 0.007 0.007 0.000 0.007
## H 0.008 0.008 0.009 0.007 0.008 0.008 0.007 0.000
```

```
 # partial mantel test ?wassermann
 wassermann(eucl.mat = eucl.mat, cost.mats = list(cost=cost.mat), gen.mat = gen.mat,
            plot=F)$mantel.tab
```

```
##                   model       r     p
## 1 Gen ~cost | Euclidean  0.4257  0.14
## 2 Gen ~Euclidean | cost -0.0312 0.575
```

Now rerun the simulation a further 20 steps and check again. . . .

```
 simpops <- run.popgensim(simpops, steps=20, cost.mat, n.offspring=para$n.offspring, n.ind=para$n.ind,
                          para$mig.rate, para$disp.max, para$disp.rate, para$n.allels, para$mut.rate,
                          n.cov=para$n.cov, rec="none")
#convert to genind to calculate pairwise fsts (this )
 gsp <- pops2genind(simpops, para$locs, para$n.cov)

 #calculate your genetic distance matrix e.g. fst or D
 gen.mat <- pairwise.fstb(gsp)    #around 150 times faster than mmod::pairwise_Gst_Nei !!!!!!!!!!,
 #but only works with simulated population of equal size
round(gen.mat ,3)
```

```
##       A     B     C     D     E     F     G     H
## A 0.000 0.031 0.020 0.014 0.015 0.018 0.016 0.021
## B 0.031 0.000 0.032 0.024 0.025 0.029 0.030 0.032
## C 0.020 0.032 0.000 0.016 0.017 0.020 0.021 0.025
## D 0.014 0.024 0.016 0.000 0.012 0.014 0.014 0.020
## E 0.015 0.025 0.017 0.012 0.000 0.013 0.016 0.021
## F 0.018 0.029 0.020 0.014 0.013 0.000 0.019 0.022
## G 0.016 0.030 0.021 0.014 0.016 0.019 0.000 0.018
## H 0.021 0.032 0.025 0.020 0.021 0.022 0.018 0.000
```

```r
# partial mantel test ?wassermann
wassermann(eucl.mat = eucl.mat, cost.mats = list(cost=cost.mat), gen.mat = gen.mat,
           plot=F)$mantel.tab
```

```
##                    model       r     p
## 1 Gen ~cost | Euclidean  0.8973 0.002
## 2 Gen ~Euclidean | cost -0.4738  0.89
```

You can now "play" with the simulator using different landscape, number of subpopulations, different locations, number of alleles, number of loci etc. For example rerun your analysis for only 4 subpopulations. How does this affect your ability to detect an effect of the landscape?