dartR Workshop - CBA

Bernd Gruber & Arthur Georges

2018-11-29

Contents

Landscape genetics

This tutorial is slightly different from the previous ones as it will guide you through an example and explains you how to create a simple resistance layer. Then you are encouraged to explore the approach using your own data or the example data set provided.

Guided example: A landsape genetic analysis

The idea of a landscape genetic analysis is that genetic similarity is between individuals/populations is dependent on the distance between individuals and [potentially] on the "resistance" of the landscape between individuals/populations.

For this example we first load a data set called possums.gl, which is already in genlight format and comes with the packages.

```
library(dartR)
```

```
possums.gl
```

```
##  /// GENLIGHT OBJECT /////////
##
##  // 300 genotypes,  200 binary SNPs, size: 499 Kb
##  0 (0 %) missing data
##
##  // Basic content
##    @gen: list of 300 SNPbin
##    @ploidy: ploidy of each individual  (range: 2-2)
##
##  // Optional content
##    @ind.names:  300 individual labels
##    @loc.names:  200 locus labels
##    @pop: population of each individual (group size range: 30-30)
##    @other: a list containing: xy  latlong
```

Task

1. Study the possum genlight object (how many individuals per population)
2. Overall how many loci are in the data set?

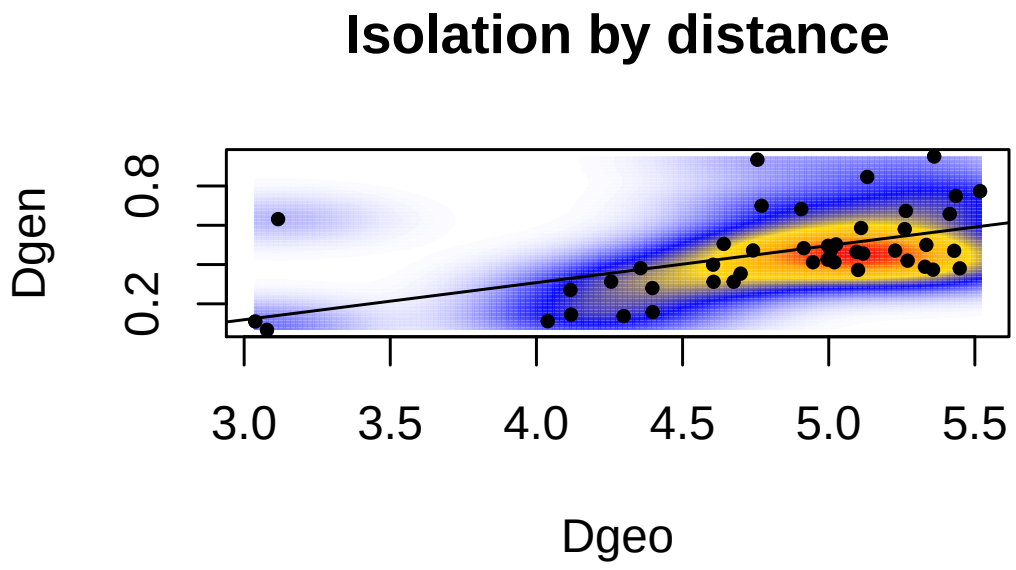
Isolation by distance

The "null model" in landscape genetics is that there is a simple relationship between genetic and euclidean distance. A standard procedure is to study the relationship between log(euclidean

distance) and Fst/1-Fst (see ?gl.ibd for details). For a quick check we can use the `gl.ibd`. To be able to use the function the genlight object needs to have the coordinates for each individual in the `@other$latlong` slot. Further we need to provide information if the coordinates are already projected or given in lat/lon.

```
iso <- gl.ibd(possums.gl, projected = TRUE)
```

```
## Standard analysis performed on the genlight object. Mantel test and plot will be Fst/1-Fst versus log(dist
## Coordinates not transformed. Distances calculated on the provided coordinates.
## Mantel statistic based on Pearson's product-moment correlation
##
## Call:
## mantel(xdis = Dgen, ydis = Dgeo, permutations = 999, na.rm = TRUE)
##
## Mantel statistic r: 0.5513
##       Significance: 0.001
##
## Upper quantiles of permutations (null model):
##   90%   95% 97.5%   99%
## 0.240 0.315 0.364 0.417
## Permutation: free
## Number of permutations: 999
```



The function returns a list of the following components: Dgen (the genetic distance matrix), Dgeo (the Euclidean distance matrix), mantel (the statistics of the mantel test)

A mantel test is basically a simple regression, but the significance takes the non-independence of pairwise distances via a bootstrap approach into account.

Landscape genetics using a landsacpe restistance approach

Often ecologists want to know if a particular landscape feature is affecting population structure on top of Euclidean distance. The idea is that a particular feature is causing some cost for individuals when moving through it, hence modifying the actual euclidean distance between individuals/populations. Commonly used approaches to calculate so-called cost-distances are the "least-cost" and "circuitscape" approach. Both approaches require a landscape that represents landscape features in terms of their "resistance" values.

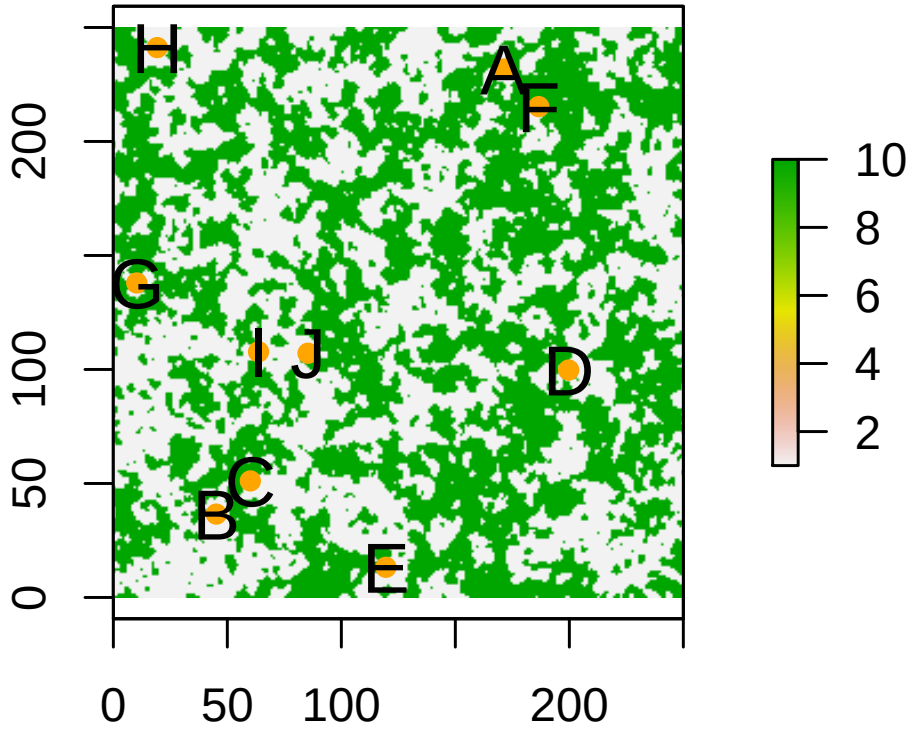
Calculation of cost distances

In this example we use populations as the entity of interest. Hence we need to calculate three population-based pairwise distance matrices, namely a Euclidean distance matrix, a cost distance matrix and finally a genetic distance matrix. The two distance matrices can then be used (very similar to the partial mantel test above) to "compete"" against each other - how well they explain genetic distances. As mentioned we base our analysis on populations, therefore we first need to calculate the coordinates of our population centers. But first we load our (resistance) landscape to be able to plot our data.

```
landscape.sim <- readRDS(system.file("extdata",
    "landscape.sim.rdata", package = "dartR"))
```

We calculate the population centers via:

```
xs <- tapply(possums.gl@other$latlong[, "lon"],
    pop(possums.gl), mean)
ys <- tapply(possums.gl@other$latlong[, "lat"],
    pop(possums.gl), mean)

plot(landscape.sim)
points(xs, ys, pch = 16, col = "orange")
text(xs, ys, popNames(possums.gl), col = "black",
    cex = 1.5)
```




```
coords <- cbind(xs, ys)
```

Euclidean distance

```
eucl <- as.matrix(dist(coords))
```

Task

1. Explore your Euclidean distance matrix.
2. Note the Euclidean distance between A-F, and I-J.

Costdistances

```
cost <- gl.costdistances(landscape = landscape.sim,
    locs = coords, method = "leastcost", NN = 8)
```

Task

1. Explore your Euclidean distance matrix.
2. Note the cost distance between A-F, and I-J and compare it to the Euclidean distances noted above. Can you see the effect of the landscape between A-F compared to I-J?

Genetic distance

For simplicity we will use pairwise Fsts between population here

```
gd <- as.matrix(as.dist(gl.fst.pop(possums.gl,
    nboots = 1)))
```

And finally run a partial mantel test

```
library(PopGenReport)
```

```
## Loading required package: raster
```

```
## Loading required package: sp
```

```
##
## Attaching package: 'raster'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
wassermann(gen.mat = gd, eucl.mat = eucl, cost.mats = list(cost = cost),
    plot = F)
```

```
## $mantel.tab
##                    model       r     p
## 1 Gen ~cost | Euclidean  0.4945 0.024
## 2 Gen ~Euclidean | cost -0.2502 0.843
```

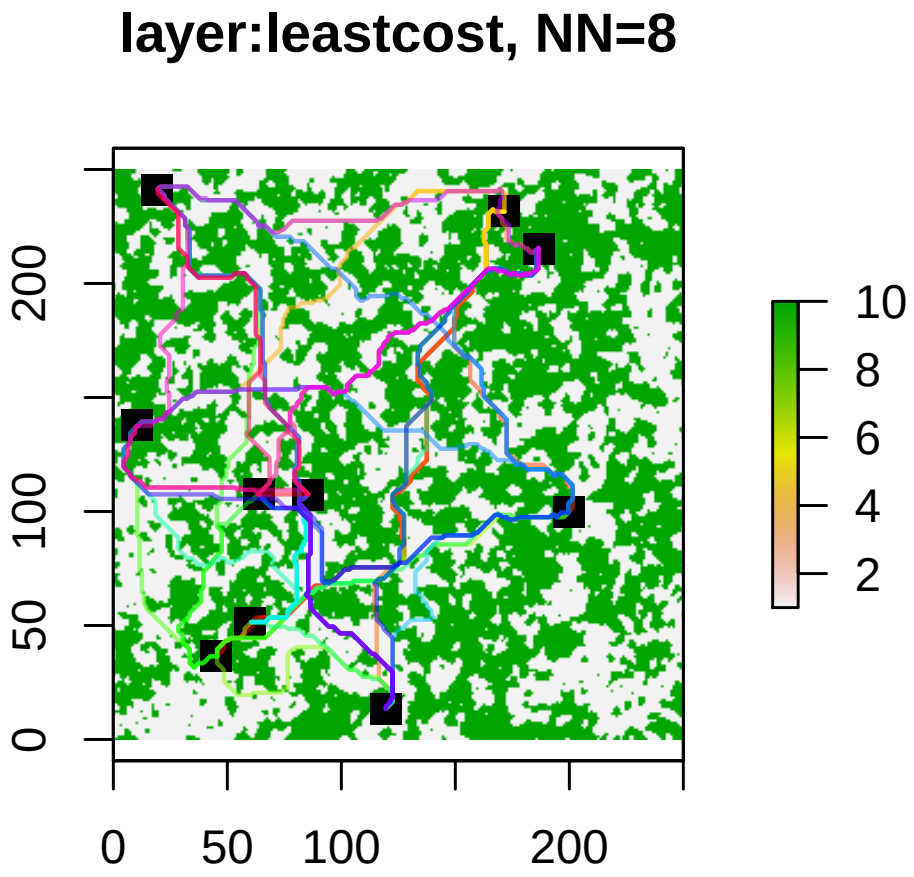
```
lgrMMRR(gen.mat = gd, cost.mats = list(cost = cost),
    eucl.mat = eucl)
```

```
## $mmrr.tab
##       layer   coefficient tstatistic tpvalue
## 2      cost  0.0014649146   3.686690   0.058
## 3 Euclidean -0.0009384741  -1.674577   0.313
## 1 Intercept  0.1228057756   4.419399   1.000
##    Fstat Fpvalue        r2
## 2 24.532   0.001 0.5387859
## 3     NA      NA        NA
## 1     NA      NA        NA
```

A landscape genetics analysis in a single command

dartR has a convinience function that does it all in once, but is less flexible. It has the benefit that it shows the actual least cost path in the landscape (but runs slower).

```
glc <- gl.genleastcost(possums.gl, fric.raster = landscape.sim,
    gen.distance = "D", NN = 8, pathtype = "leastcost")
```




```
wassermann(gen.mat = gd, eucl.mat = eucl, cost.mats = list(cost = cost),
    plot = F)
```

```
## $mantel.tab
##                    model       r     p
## 1 Gen ~cost | Euclidean  0.4945 0.025
## 2 Gen ~Euclidean | cost -0.2502 0.841
```

Run your own simulated resistance surface landscape analysis

An excellent way to test an approach and understand its limitation, is to use a simulation appraoch. You will be provided with functions to:

A. simulate a landscape (a resistance layer) B. Populate this landscape with populations of

a simulated species C. "Run" the species over the landscape for a number of generations D. Perform a landscape genetic analysis using a resistance layer.

Then you can go back and explore the parameters and its effect on the performance of the approach (e.g. change the number of loci, change the number of generations etc.)

We need to install another package: `secr`.

```
install.packages("secr")
library(secr)
library(PopGenReport)  #should already be installed
library(raster)  #should already be installed
```

Before we can start we also need to download some additional functions that simplifies the coding and scripts.

```
source("https://raw.github.com/green-striped-gecko/lgfun/master/lgfuns.r")
```

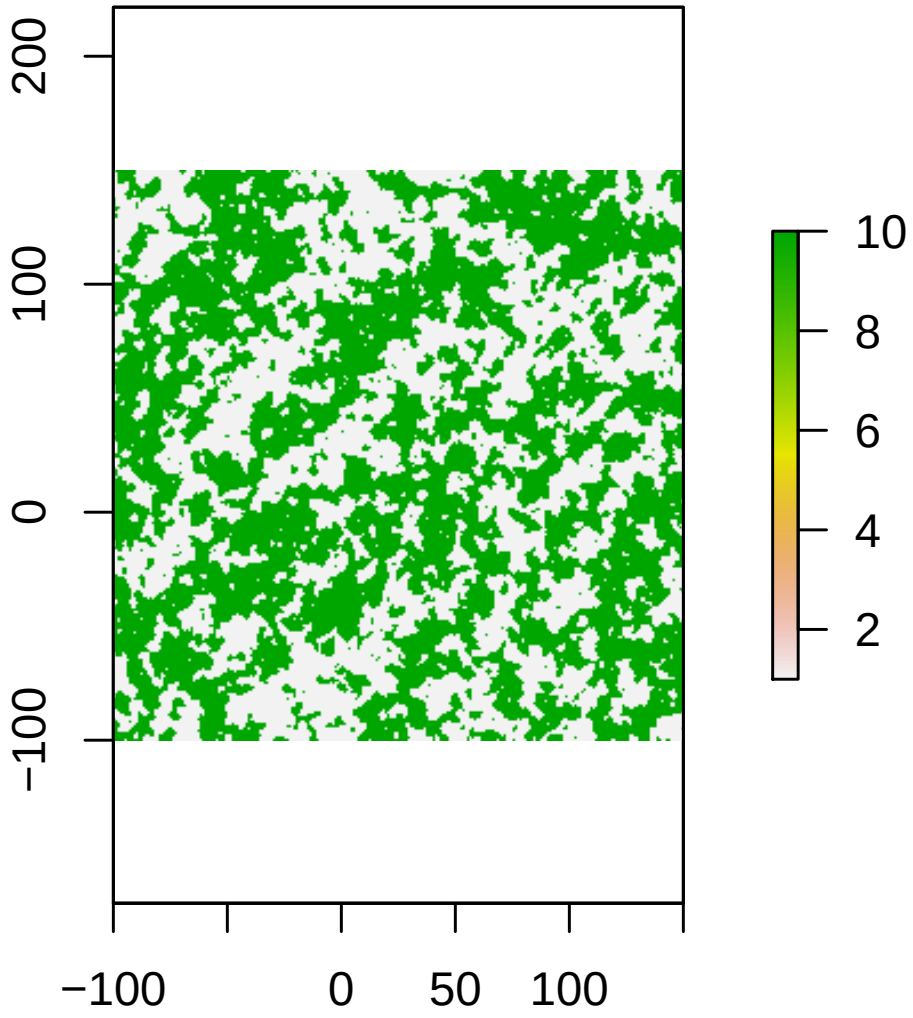
A. Simulate a resistance layer

We will use the 'randomHabitat' function from the 'secr' package to create a resistance layer, but you could simply load a png file or any other image file format with the 'raster' function from package 'raster' (?raster::raster, see the examples in there).

Parameter A is the amount of habitat and p controls the amount of clumping. Nx, ny determine the extend of the landscape and resVal determines the resistance values of non-Habitat features.[1]

[1] Check ?randomHabitat for details

```
r <- create.resistance(nx = 50, ny = 50, p = 0.5,
    A = 0.5, resVal = 10)
```


200
100
0
−100
−100
0
50
100
10
8
6
4
2

Task

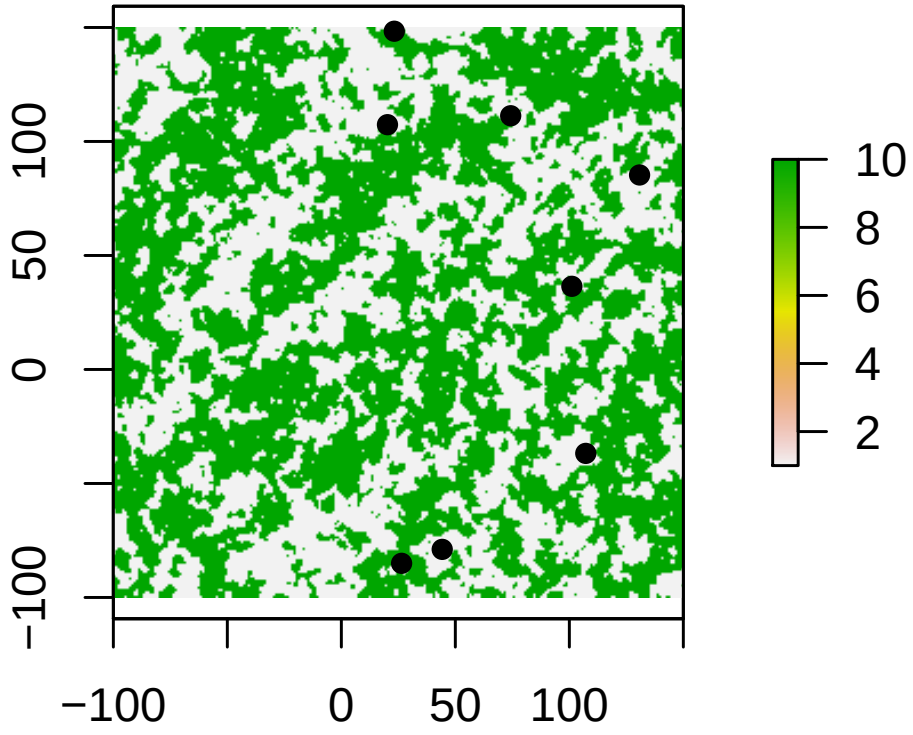
1. Rerun the command above until you "like" the resistance layer. You can change ther parameters (but be aware that p should not be higher than 0.5, otherwise the algorithm does not work reliably)

B. Populate the landscape with population of a species

B.1. Add populations to the landscape (using minimal distance)

`createpops` allows us to set up 'n' subpopulations in the habitat only (non green areas). The subpopulations should be at least 'mindist' units apart.

```
locs <- create.pops(n = 8, mindist = 3, landscape = r,
    plot = TRUE)
```




```
locs
```

```
##          lx        ly
## 1 130.80329  85.30945
## 2  26.42824 -85.02756
## 3 107.21551 -36.80228
## 4  74.28119 111.28365
## 5  44.24256 -78.90674
## 6  23.21800 148.35925
## 7 101.11774  36.46223
## 8  20.23693 107.36946
```

B.2. Initialise a metapopulation

We use 'init.popgensim' from package PopGenReport 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

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!!)
```

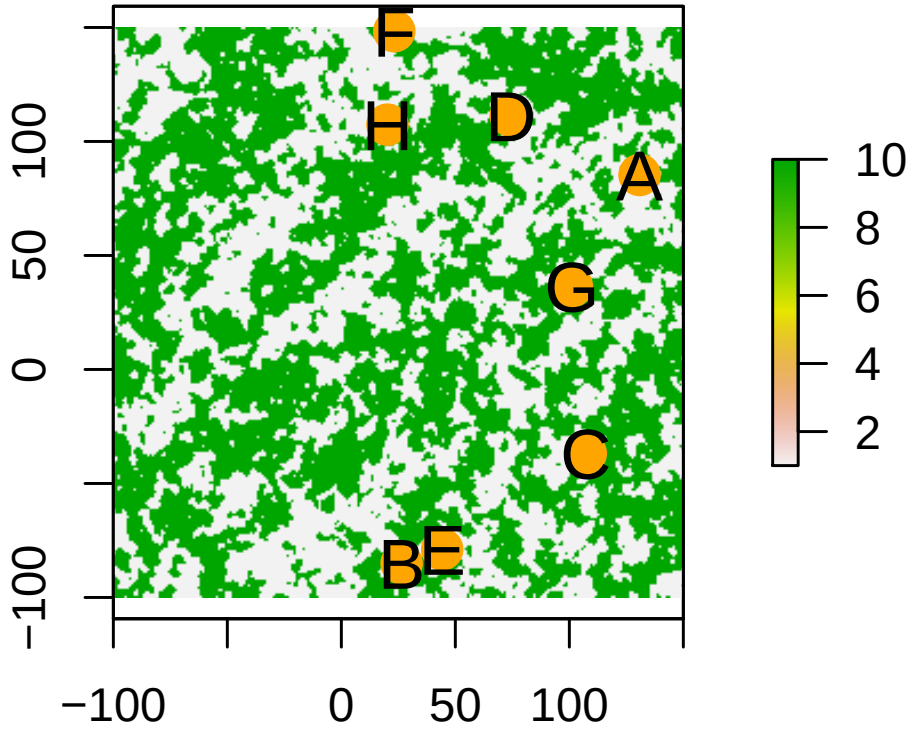
Define population dynamics:

```
# 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 [create 100 SNPs]
para$n.allels <- 2
para$n.loci <- 100
para$mut.rate <- 1e-05
```

Define a cost distance method:

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

# Initialize simulation of populations from
# scratch
```

```
landscape <- r  #<-raster(system.file('external/rlogo.grd', package='raster'))

# Define x and y locations
para$locs <- locs
# give the population some names
rownames(para$locs) <- LETTERS[1:para$n.pops]


# Create a costdistance matrix

cost.mat <- gl.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)
```




```
# 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] 2
##
## $n.loci
## [1] 100
##
## $mut.rate
## [1] 1e-05
##
## $method
## [1] "leastcost"
##
## $NN
## [1] 8
##
## $locs
##          lx        ly
## A 130.80329  85.30945
## B  26.42824 -85.02756
## C 107.21551 -36.80228
## D  74.28119 111.28365
## E  44.24256 -78.90674
## F  23.21800 148.35925
## G 101.11774  36.46223
## H  20.23693 107.36946
```

B.3 Initialise your population on the landscape

Now finally we can initialise our population using the init function

```
simpops <- init.popgensim(para$n.pops, para$n.ind,
    para$sex.ratio, para$n.loci, para$n.allels,
    para$locs, para$n.cov)
```

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"
```

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 genlight object, then we can use our usual functions (e.g `gl.fst.pop`).

```
glsp <- pops2gl(simpops, locs = para$locs)
```

```
## Loading required package: parallel
```

```
glsp  #check the genlight object
```

```
##  /// GENLIGHT OBJECT /////////
##
##  // 800 genotypes,  100 binary SNPs, size: 1.2 Mb
##  0 (0 %) missing data
##
##  // Basic content
##    @gen: list of 800 SNPbin
##    @ploidy: ploidy of each individual  (range: 2-2)
##
##  // Optional content
##    @ind.names:  800 individual labels
##    @loc.names:  100 locus labels
##    @pop: population of each individual (group size range: 100-100)
##    @other: a list containing: xy
```

```
summary(glsp)
```

```
##   Length    Class     Mode
##        1 genlight       S4
```

```
gen.mat <- as.matrix(as.dist(gl.fst.pop(glsp,
    nboots = 1)))
# overall pairwise fst
mean(gen.mat)
```

```
## [1] 3.72438e-05
```

Task

1. Check the pariwise Fsts between populations. Why are they so low? Is there an effect of the landscape on the population structure, yet?
2. Run a partial mantel test and check the result []

Now we run our simulation by simply passing our simpops, with some additional parameters that are needed for the simulation. The number of generations 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).

*C.Run simulations and analyze results***

C.1. 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 the population structure?

We can check now the pairwise Fst values and then do a landscape genetic analysis using partial mantel tests.

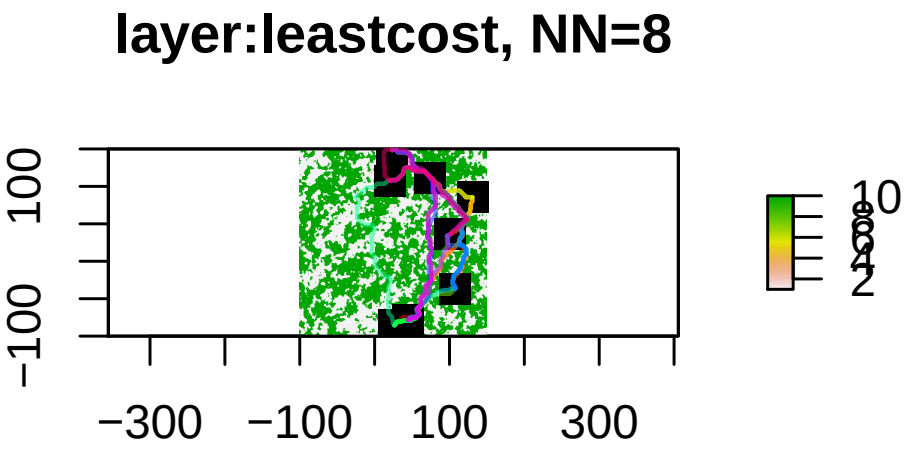
Task

1. Check the pariwise Fsts between populations after the 3 generation have passed. Is there now an effect of the landscape on the population structure, yet?
2. Run a partial mantel test and check the result
3. Run now your population another 50 generation.

simpops <- run.popgensim(simpops, steps=50,….)
How did the overall mean pairwise Fst change?
Is there now a significant effect of the landscape on the population structure.

Suggestion

Create a loop and calculate mean pairwise Fst values every 10 generations (up to 200) and colour the dots by the significance of the Mantel test.



Task

You can now "play" with the simulator using different landscapes, number of subpopulations, different locations, 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? Or you can try to use your own data and create a resistance layer. Be aware that the coordinate system between your landscape and genlight object needs to be the same preferably projected coordinate system.

Create a resistance layer for your own data (experimental!!!)

Admittedly this is a bit of a stretch, as originally you would like to have a gis data set with landscape features to create a resistance layer. Here will will use a "quick and dirty" approach and create a resistance landscape from open source maps, which may be not too useful for your data set.

The idea of this tutorial is to demonstrate the approach for an example of our fox data

Make sure you have installed two additional libraries:

```
# install via: library(devtools)
# install_github('dkahle/ggmap')
library(ggmap)
library(dartR)
library(osmdata)
# also load some helper functions

source("https://raw.github.com/green-striped-gecko/lgfun/master/lgfuns.r")



## define your genlight data set here we use
## the foxex in NSW
gl <- foxes.gl[pop(foxes.gl) == "NSW", 1:100]

## bounding box in lat/lon
ll <- gl@other$latlong
bb <- matrix(c(min(ll$lon), min(ll$lat), max(ll$lon),
    max(ll$lat)), nrow = 2, ncol = 2)

# or you can use geocode functions bb <-
# getbb('South Australia')

# get a map you may need to play with zoom to
# get a suitable map also you could try
# terrain-lines to get roads if your study is
# close to a city
map <- get_stamenmap(bb, zoom = 6, maptype = "terrain-background")
plot(map)
```

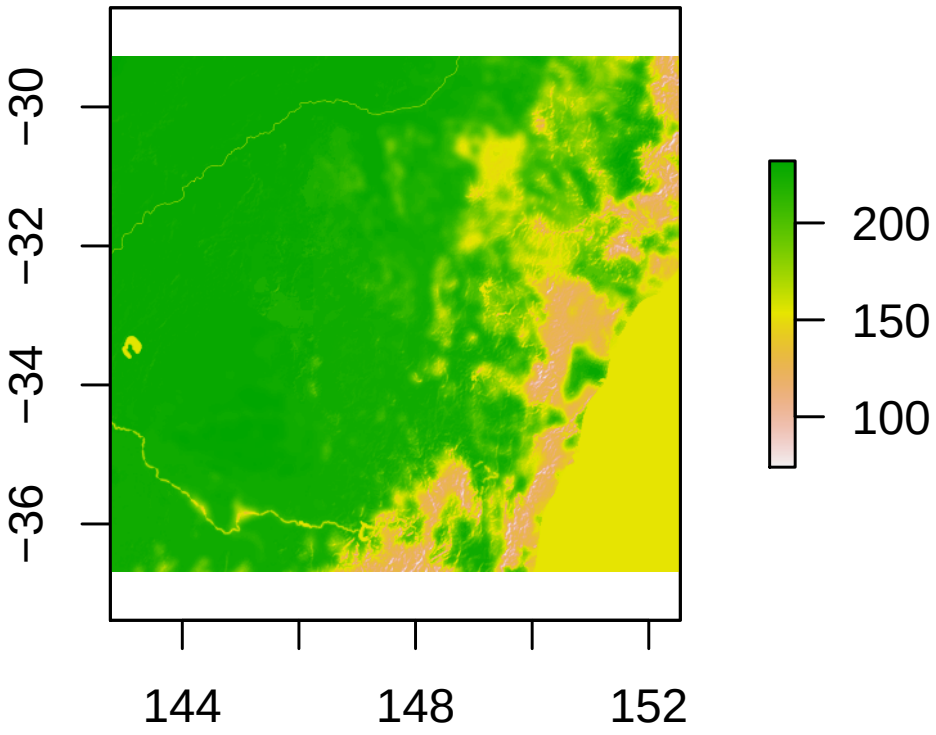
We then need to convert the image to a raster

```
## converts the image to a raster (only the red
## values are returned)
map.raster <- ggmap_to_raster(map)[[1]]
map.raster
```

```
## class       : RasterLayer
## dimensions  : 402, 444, 178488  (nrow, ncol, ncell)
## resolution  : 0.02201689, 0.01840868  (x, y)
## extent      : 142.7657, 152.5412, -36.6801, -29.27981  (xmin, xmax, ymin, ymax)
## coord. ref. : NA
## data source : in memory
## names       : layer.1
```

```
## values      : 74, 232  (min, max)
```

```
plot(map.raster)
```



Obiously this is not a very suitable map.

‘‘‘