# Optimizer

*Gruber, Hemming-Schroeder, Otero-Jimenez, Ramsay, Ricci, & Balkenhol*

*November 21, 2016*

## Opt.landgen: A function in the PopGenReport package for optimizing a sampling design to detect landscape effects on gene flow

### Background

The main goal of the opt.landgen function in PopGenReport is to find an optimized sampling design to test if a resistance surface has a significant effect on population genetic structure in a given study system. Though landscape geometry with respect to populations substantially affects the ability to find a significant association between landscape features and population structure, this aspect has so far received little attention. The power to detect the genetic to landscape relationship depends on the difference between euclidean and landscape cost distances. The opt.landgen function works by detecting sampling designs that maximize the variance in the variance of detours (where detour is the difference between cost distance and euclidean distance) following the "Newman allocation" principle. This approach implemented in the opt.landgen function has been validated using a simulation approach which demonstrated that the optimized sampling design had increased power in detecting an effect of the landscape where a landscape effect existed (Gruber et al 2017).

In part I of this tutorial, we demonstrate how to use the opt.landgen function to design a sampling scheme. In this section, we also show how to use two features that may increase the utility of the opt.landgen function. First, we demonstrate how to use a mask on the landscape for if there are areas within the landscape of known non-habitat, so that populations in these areas are not selected. Second, we show how to fix a portion of the populations for if the opt.landgen function is being used to add additional populations after some have already been sampled. In part II of this tutorial, we show how to use the opt.landgen function post-hoc to test the relative power of a fixed sampling design.

### Set up

To use the opt.landgen function, you will need to install the PopGenReport package. For complete instructions on how to install and use PopGenReport, please refer to the PopGenReport tutorial available at: https://cran.r-project.org/web/packages/PopGenReport/vignettes/PopGenReportIntroduction.pdf or to http://www.popgenreport.org

Load necessary packages

```
library(PopGenReport)
library(hierfstat)
```

### Part I: Using opt.landgen to find an optimized sampling design

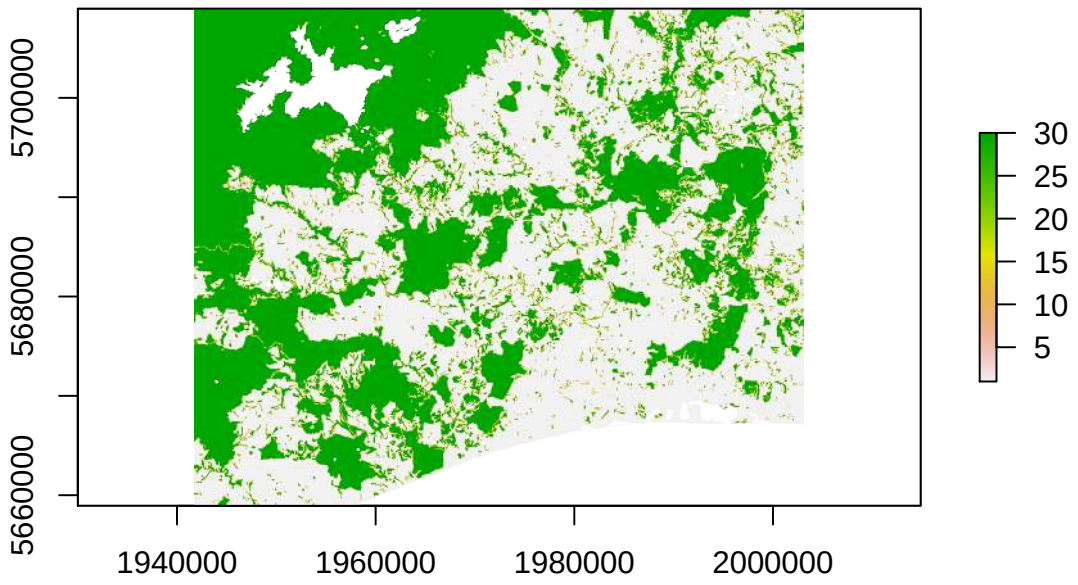
#### Load the resistance surface

First we need to load our map and plot it. You can use already created cost surface or create a cost surface from any raster file. The `raster` function from the raster package allows to virtually load any type of raster file (including georeferenced tiff files, ascii files, ESRI raseter)

types of files that can be used???

Here we load a layer of the proportion of treescrub in the New Zealand study area and rescale it so that cells with 100% treescrub have no cost and areas with 0% treescrub have a high cost. The example is based on a study on the effect of tree/scrub cover on gene flow in possums at Hawke Bay, New Zealand (Sarre et al. 2014)

```
# load the raster
treeScrub <- raster("treeScrub.asc")

# the current raster shows the proportion of each cell that is treescrub
# rescale it so that cells with 100% treescrub will have a resistance of 1
# and cells with no treescrub have a a resistance of 30 because of the costs
# of moving, an cost of 1 (which basically is the equivalent to isoloation
# by euclidean distance)
forest <- 1 + 29 - treeScrub * 29
plot(forest)
```




```
# rescale the waterbodies (from NA to 1000) removing of NAs as no mask is
# used.
forest[is.na(forest[])] <- 1000
```

### Use the optimizer

In addition to the resistance surface, the optimizer requires two pieces of information.

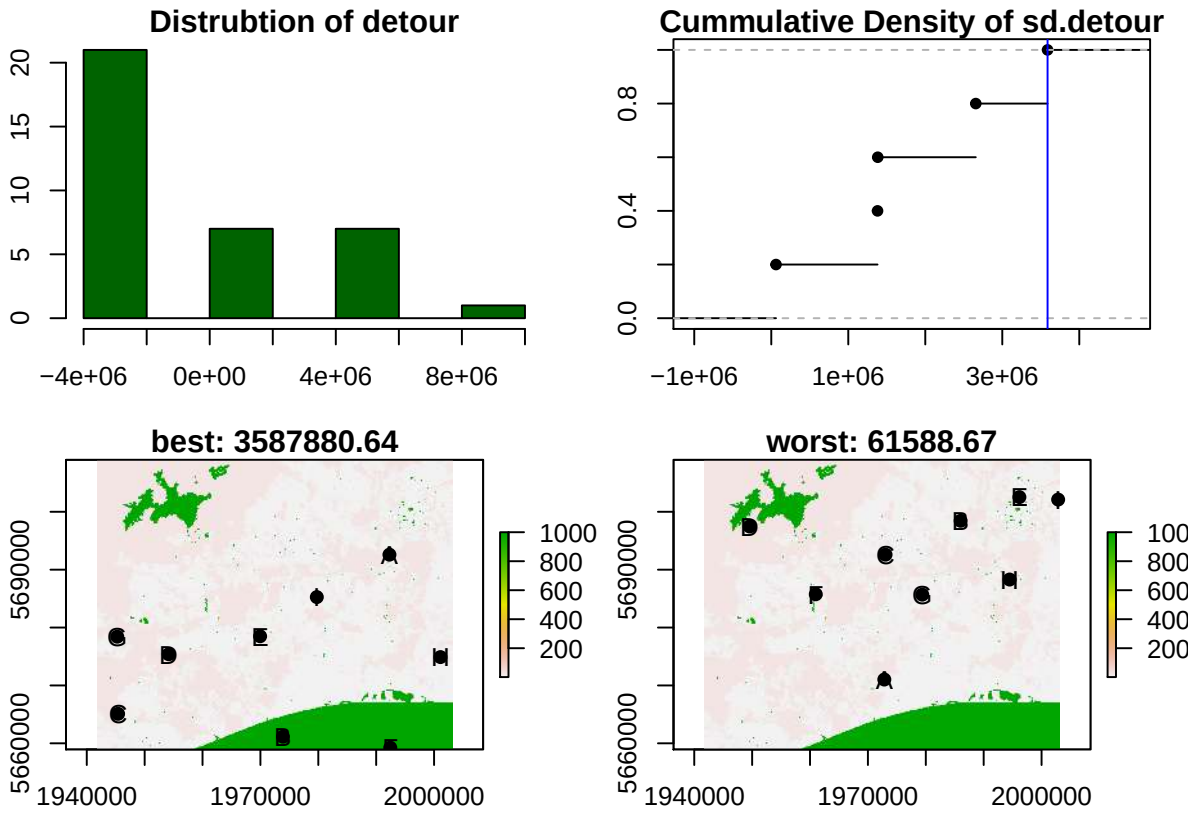
- **nlocations**: the number of sampling locations to be optimized
- **method**: the method of calculating the cost distance based on the gdistance package (van Etten 2014) costdistance function. There are currently three types of cost distances implemented: "leastcost", "commute", "SPDistance". For an explanation of these options refer to: `?landgenreport` and Gruber & Adamack (2015). The following arguments have default values that can be changed to better suit your sampling scheme.
- **mindist**: the minimum distance between sampling locations in map units (default = 0)
- **NN**: the number of next nearest neighbor cells to be used. See function costdistance (default = 8)
- **iter**: the number of iterations that should be used to find an optimised design (default = 100), but it is advisable to set the number to higher values, so the optimizer is able to find a near to optimal design (order of 1000 to 10.000). Due to the potentially prohibitiv possible sampling design, it is quite unlikely to find the optimal design if the sampling space is not severely restricted (e.g. by mask or mindist)
- **retry**: the number of retries if optimisation is not possible in a certain iteration due to mindist and/or fixed locations (default = 10). An output is created to make the user aware if the algorithm has difficulties to find suitable sampling designs.

Additional information can be added if some sampling locations are predetermined or not all areas of the cost surface are available for sampling.

- **fixed**: a data.frame of n fixed sampling locations with dimensions n x 2
- **mask**: a raster object that masks positions which are not available. Areas which are not to be used for locations are coded as NA (not available), all other values are treated as suitable for placing locations
- **plot** if TRUE a graphical output is produced (description see below)

In this example we optimize 9 sampling locations on the forest cost surface using the least cost method.

```
# set the seed of the number generator to achieve the same output.
set.seed(123)
op <- opt.landgen(landscape = forest, nlocations = 9, mindist = 5000, method = "leastcost",
    NN = 8, iter = 5)
```




In order to expedite the function, we set iter = 5, which will run for 1-2 minutes. If you set iter = 100, it will run for about 10 minutes and produce a much better comparison. For a productive runs we suggest to use values of at least iter=1000 to 10000.

**Output**

```
op
```

```
## $opt
##      sd.cost  sd.detour
## 3 3591386.34 3587880.64
## 4 2655980.97 2655712.15
## 2 1384178.83 1382888.10
## 1 1414872.33 1380046.65
## 5   77889.87   61588.67
##
## $scenario
## $scenario[[1]]
##        xs      ys
## A 1992293 5692594
## B 1973893 5661151
## C 1945411 5665099
## D 1954153 5675392
## E 1969945 5678494
```

```
## F 1992434 5659247
## G 1945270 5678423
## H 2001106 5674898
## I 1979744 5685262
##
## $scenario[[2]]
##        xs      ys
## A 1993351 5679269
## B 1945129 5694850
## C 1966208 5701970
## D 1983974 5660657
## E 1988768 5664182
## F 1978193 5674546
## G 1981154 5669611
## H 1991306 5707963
## I 1988204 5685896
##
## $scenario[[3]]
##        xs      ys
## A 1953800 5686108
## B 1950064 5661010
## C 1999766 5686813
## D 1993210 5674969
## E 2002163 5680609
## F 1970438 5704367
## G 1990742 5664112
## H 1977206 5697317
## I 1964939 5707046
##
## $scenario[[4]]
##        xs      ys
## A 1952813 5694568
## B 1974880 5669258
## C 1983410 5688293
## D 1959370 5664323
## E 1973117 5661856
## F 1969099 5706976
## G 1970861 5682724
## H 1998427 5665169
## I 1978616 5682442
##
## $scenario[[5]]
##        xs      ys
## A 1972835 5671021
## B 1986019 5698445
## C 1973047 5692594
## D 1949570 5697458
## E 1996171 5702534
## F 1960991 5685755
## G 1979392 5685685
## H 1994479 5688293
## I 2002868 5702111
```

The opt.landgen function produces a list containing the standard deviation of the least cost values (sd.cost)

and the standard deviation of the detours (sd.detour) for each iteration [opt] as well as the location coordinates of the sampling locations [scenario].

If plot = TRUE it will also produce four plots

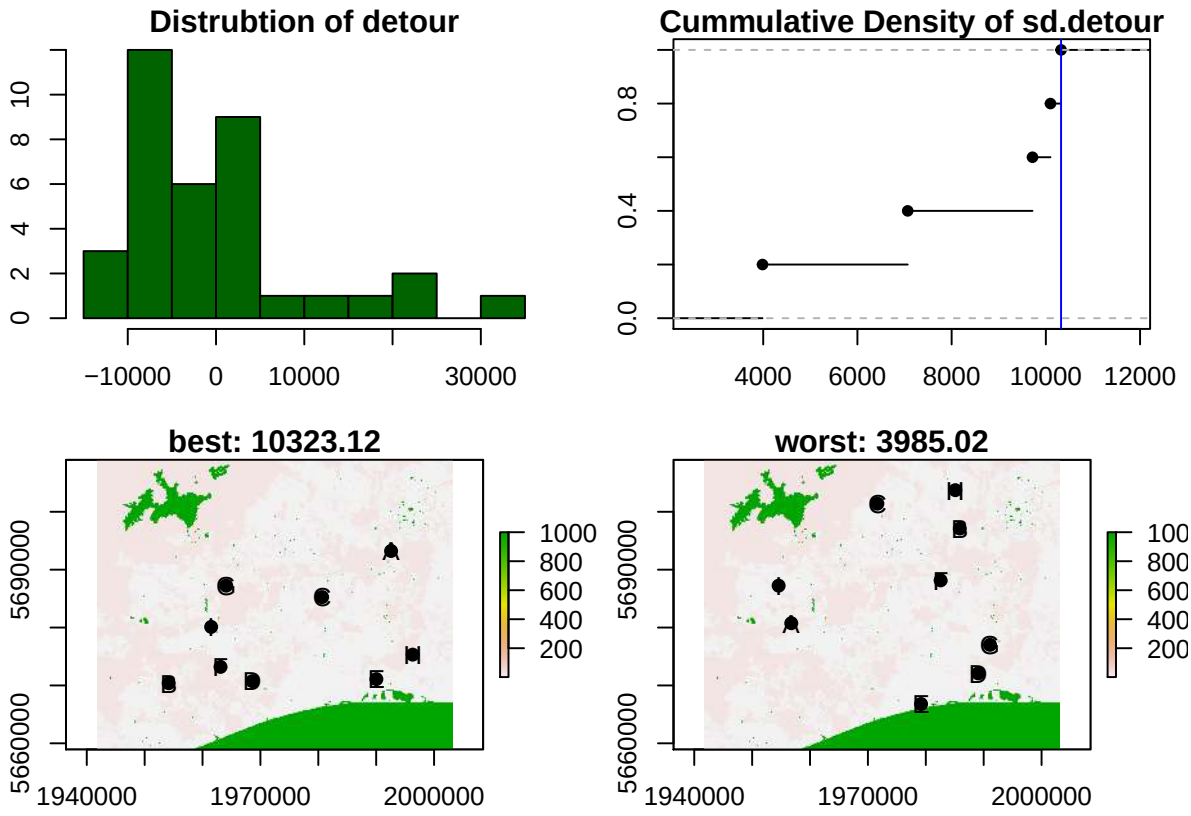
- Distribution of detour: a desirable distribution is U-shaped, which is hard to achieve. More likely a wide, platykurtic normal distribution is the best that can be achieved.
- Cummulative density of sd.detour: Shows the proportion of iterations with lower standard deviations
- Best: the coordinates of the iteration with the highest sd.detour plotted on the cost surface
- Worst: the coordinates of the iteration with the lowest sd.detour plotted on the cost surface

Because we are only going through a small number of iterations (iter=5) to save runtime in this example, it is unlikely that we have found the optimal or even a good sampling design.

### Use a mask to exclude certain areas from sampling

Suppose areas with little treescrub are known to not be inhabitat by the species or some areas are not accessible and therefore sampling in these locations is not possible. We can create a mask so that the optimizer will not include these areas as sample locations.

```
# define the mask so that cells with a proportion of treescrub less than .10
# will be not available
mask <- treeScrub
mask[values(treeScrub < 0.1)] <- NA

# run the optimizer
opmask <- opt.landgen(landscape = forest, nlocations = 9, mindist = 5000, method = "leastcost",
    NN = 8, iter = 5, mask = mask)
```




## Part II: Using opt.landgen to test the power of your sampling design post-hoc

opt.landgen can also be used to compare your sampling design to an optimized version post-hoc. For this example, we compare the sampling design of an actual sample of 9 populations of possums in New Zealand to the optimized version. The data set is publicly available and the accompanied publication can be found under: Sarre, S. D., Aitken, N., Adamack, A. T., MacDonald, A. J., Gruber, B. and Cowan, P. (2014), Creating new evolutionary pathways through bioinvasion: the population genetics of brushtail possums in New Zealand. Mol Ecol, 23: 3419–3433.For demonstration purposes we will use only a subset of the data set.

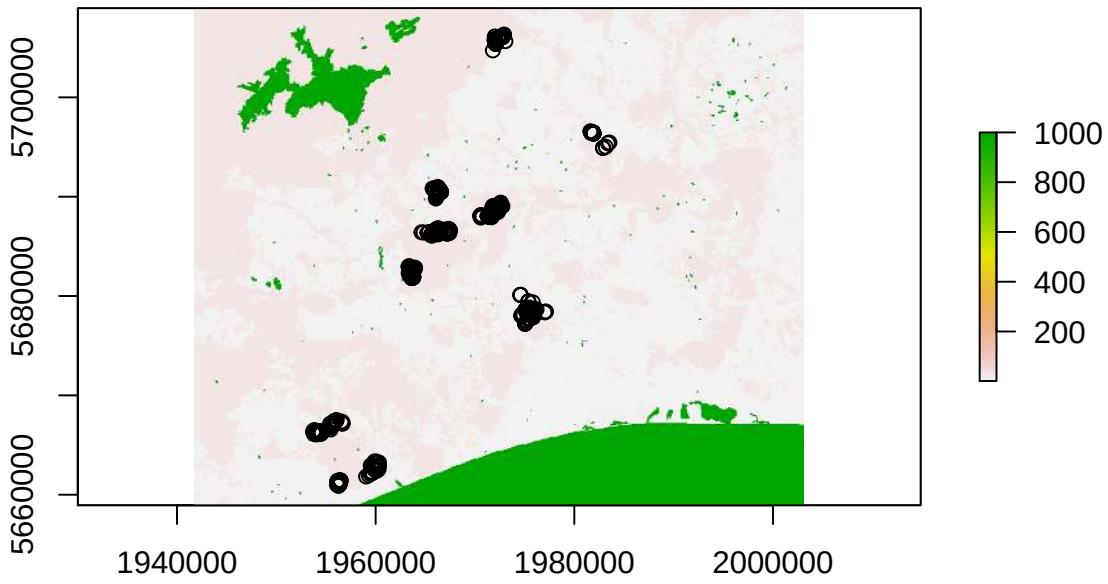
### Load the sampled location data

Sampling locations are loaded and converted into a genind object based on the adegenet package (Thombart 2008). The genind object is a very convenient data format for genetic data sets in R, but you can use any other format. You simply need to be able to provide coordinates for every sample.

```
# see ?read.genetable in PopGenReport for an explanation (creates a genind
# object for a tabular genetic data set)
possums <- read.genetable("possums.csv", ind = 1, pop = 2, lat = 3, long = 4,
    other.min = 5, other.max = 6, oneColPerAll = TRUE)

# the original dataset includes [31] populations but to expedite the
# computational time we are restricting the dataset to the first 9
# populations sampled
possums <- possums[as.numeric(pop(possums)) < 10, ]
```

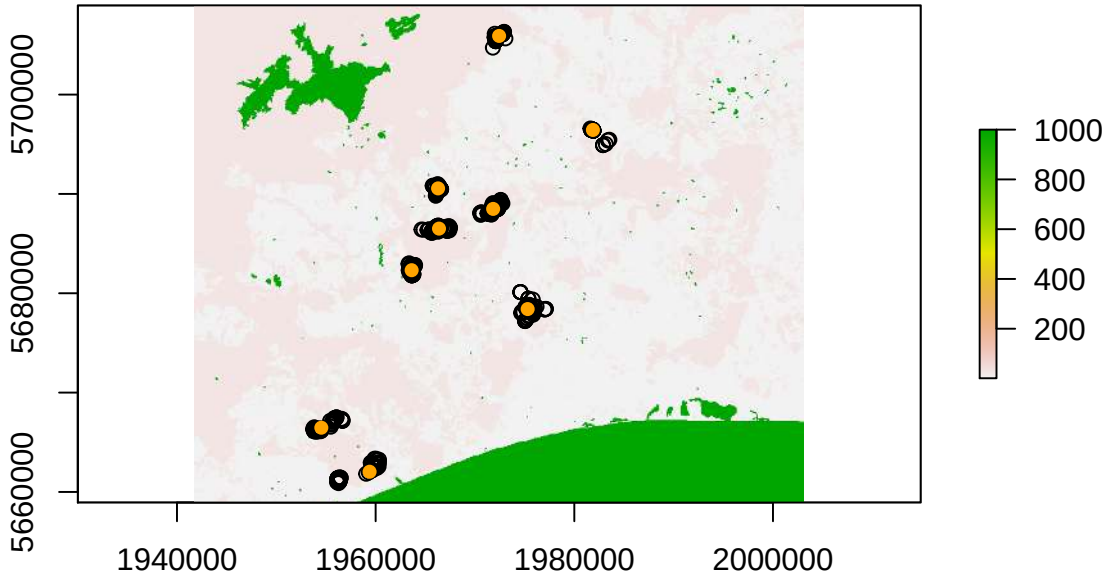
The longitude and latitude in possums is currently given in lat-lon (WGS84). Before plotting these locations on the forest layer, they must first be transformed to the New Zealand coordinate system [New Zealand Transverse Mercator 2000 (NZTM2000)], which is the coordinate system of your forest resistance layer.

```
# explain the functions used in this chunk????

pt <- possums@other$latlong
coordinates(pt) <- ~long + lat
proj4string(pt) <- "+proj=longlat +datum=WGS84 +ellps=WGS84"

nzp4 <- "+proj=tmerc +lat_0=0 +lon_0=173 +k=0.9996 +x_0=1600000 +y_0=10000000 +ellps=GRS80 +units=m +no

ptnz <- spTransform(pt, CRS(nzp4))

possums@other$xy <- coordinates(ptnz)

# plot the sampled possums on the forest layer
plot(forest)
points(possums@other$xy)
```




The sampling data is at an individual level but we are optimizing the surface for 9 populations. We can find the median center of the individuals to compare to the optimized population locations. It is obviously possible to use individuals and their coordinates instead of median centers for you sampling design, but here we use a subpopulation based design. So the intention was to sample several individuals at 9 spatially focused locations and use subpopulation based genetic distances such as Fst or Gst to study the effect of landscape structure on gene flow.

```
# find the median longitude and latitude for each population
xys <- cbind(tapply(possums@other$xy[, 1], pop(possums), median), tapply(possums@other$xy[,
    2], pop(possums), median))

# plot the sampled possums on the forest layer
plot(forest)
points(possums@other$xy)
# plot the median center locations on top of the forest layer and the
# individual possum locations
points(xys, col = "orange", pch = 16)
```

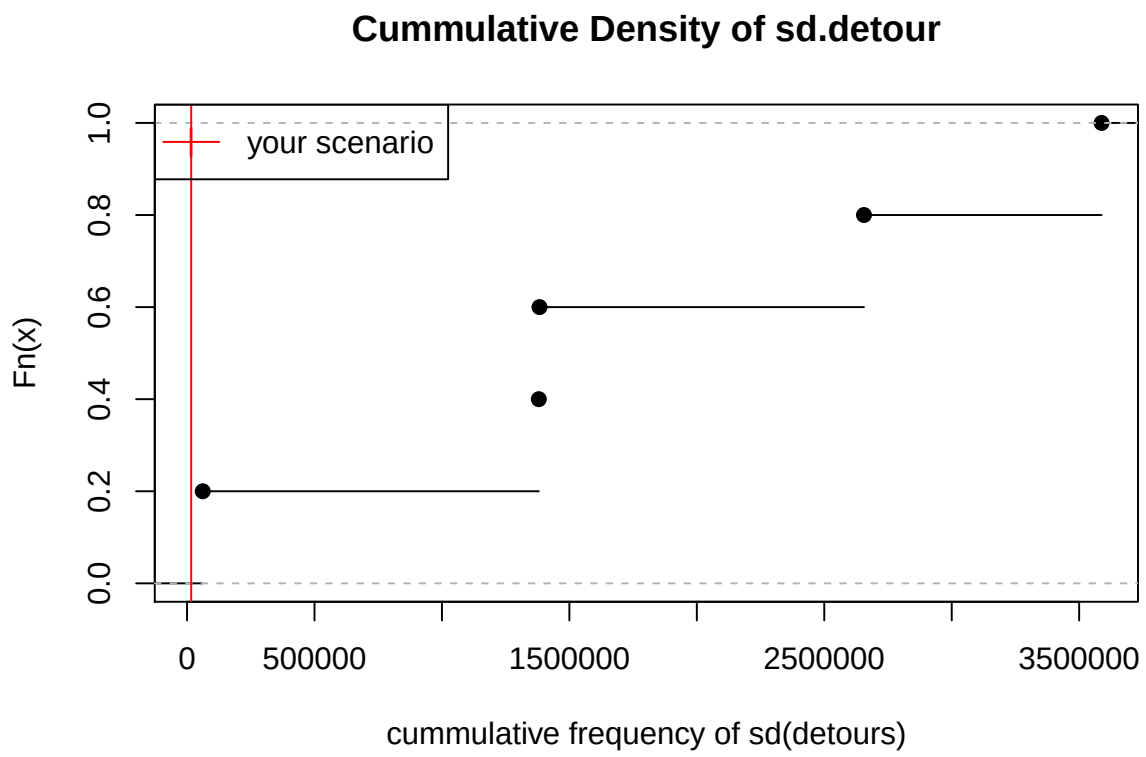



### Compare the sampled locations to the optimized locations

If we calculate the standard deviation of detours in the sampled locations, we can compare the amount of deviation to the cumulative distrobution of sd.detour calculated by the optimizer (plot 2 from the optimizer output).

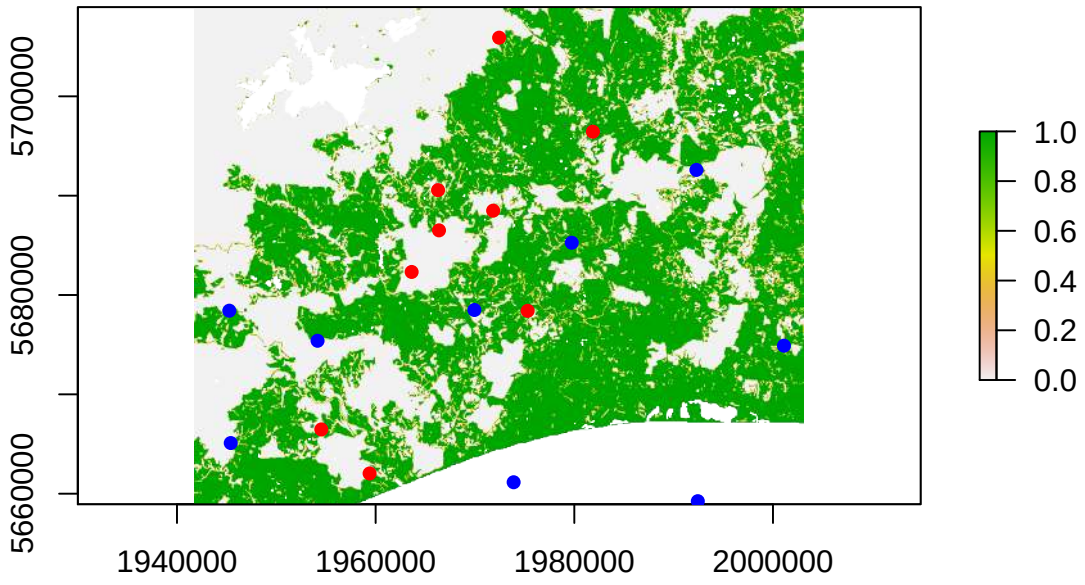
```
# cost distance between the sampled populations
cd <- as.dist(costdistances(landscape = forest, locs = xys, method = "leastcost",
    NN = 8))

# euclidian distance between the sampled populations
ed <- dist(xys)
```

```
# length of detours between sampled populations calculated as the residuals
# of the regression of cost distance by euclidian distance
detours <- resid(lm(cd ~ ed))

# the standard deviation of detours for the sampled populations
sddetours <- sd(detours)


# plot the cumulative distrobution of sd.detour with the sd.detour of our
# sampling locations overlayed

plot(ecdf(op$opt$sd.detour), main = "Cummulative Density of sd.detour", xlab = "cummulative frequency o
    xlim = range(op$opt$sd.detour, sddetours))
abline(v = sddetours, col = "red")
legend("topleft", legend = "your scenario", lwd = 1, pch = "|", col = "red")
```




The sampled and optimized scenarios can also be compared visually.

```
plot(treeScrub)
# sampled populations
points(xys, col = "red", pch = 16)
# best scenario locations
points(op$scenario[[1]], col = "blue", pch = 16)
```



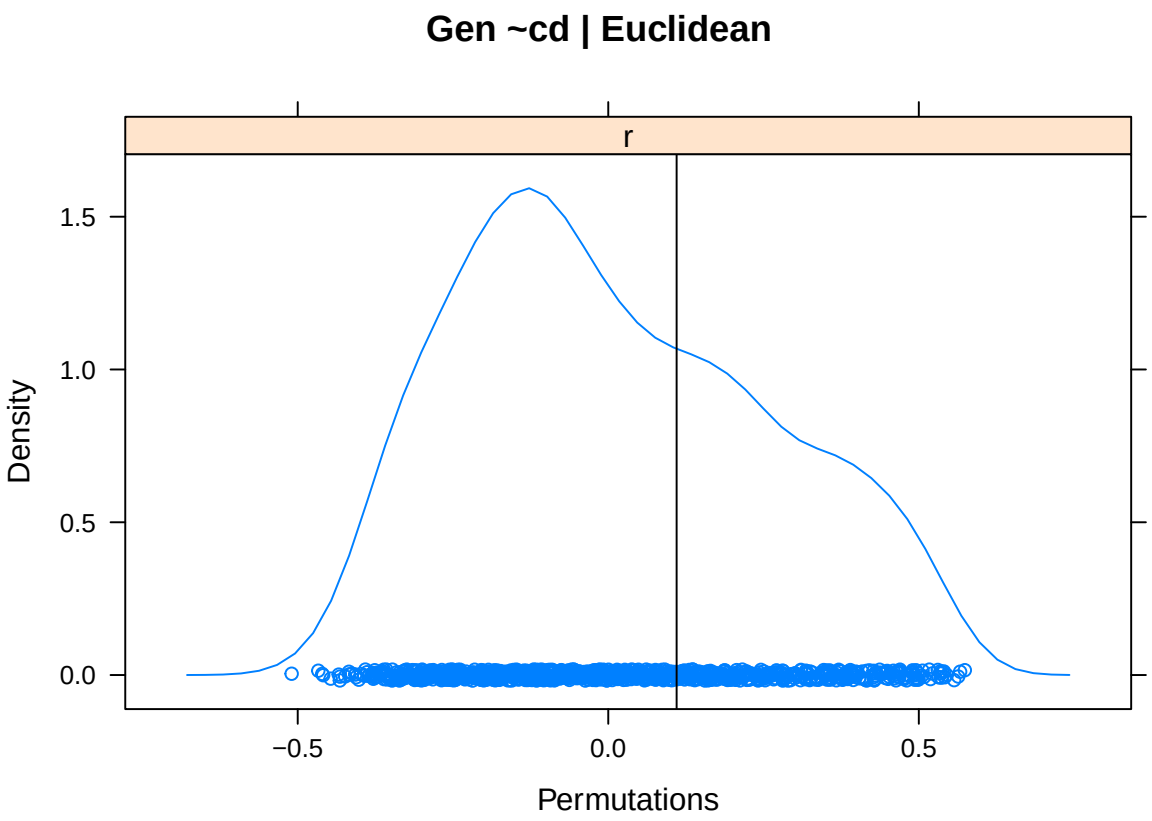

As you can see the sd.detour of the sampled locations is fairly narrow and if run long enough (iter>100) a much better sampling design could be found.

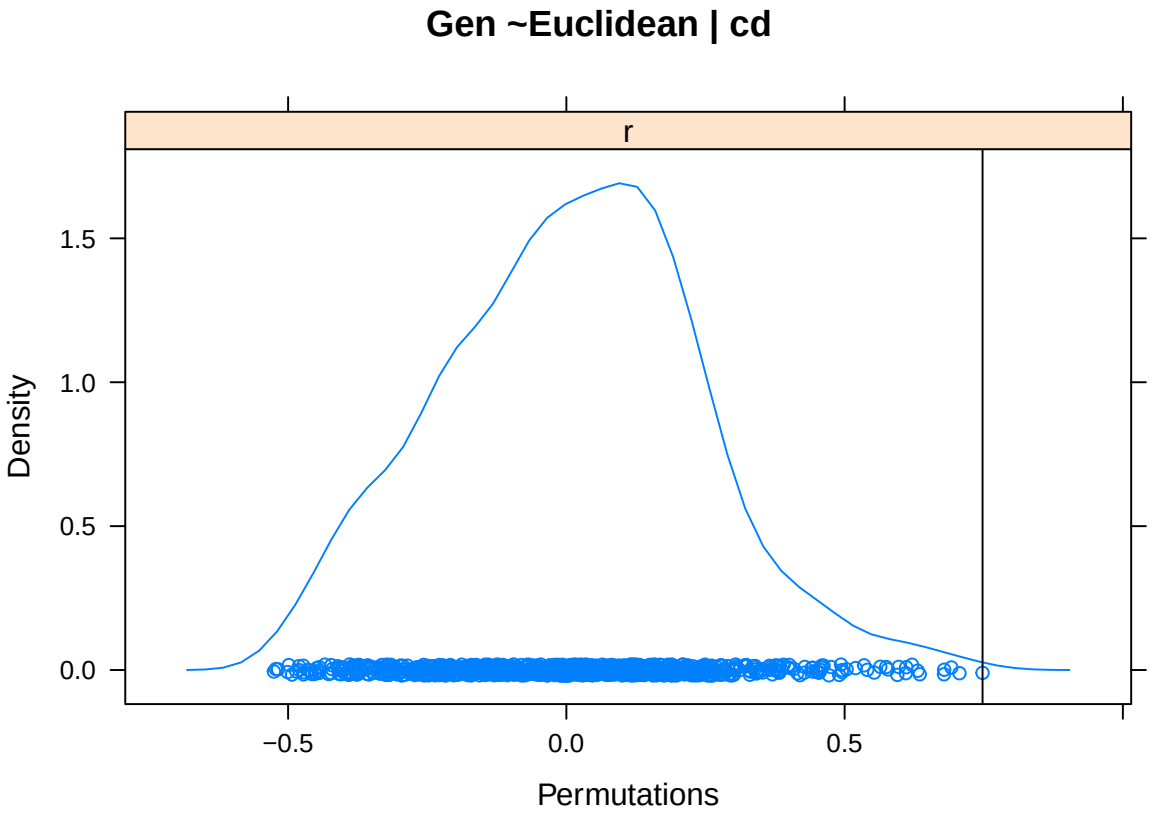
### genetic something or other

Because we have genetics data from the sampled possums, we can see how a different sampling scheme could improve our ability to find the affect of the landscape

To do so we will be using the hierfstat package to calculate pairwise Fst values between the 9 subpopulations.

```
# calculate the pairwise Gst between possum [populations or ind?]
gd <- pairwise.fst(possums)

# run a mantel test to test for the effect of the landscape (cd) on the
# genetic differences (gd) when the euclidian distance is controlled for
# (ed)
pm <- wassermann(gen.mat = as.matrix(gd), cost.mats = list(cd = as.matrix(cd)),
    eucl.mat = as.matrix(ed))
```


Gen ~cd | Euclidean
r
Density
1.5
1.0
0.5
0.0
−0.5
0.0
0.5
Permutations




```
pm
```

```
## $mantel.tab
##                  model      r     p
## 2 Gen ~Euclidean | cd 0.7479 0.001
## 1 Gen ~cd | Euclidean 0.1101 0.343
```

From the mantel test we can see that we are unable to show an effect of the density of scrubtree on the genetic distance between populations when euclidian distance is accounted for.

As a simple test we study the correlations between genetic, cost and Euclidean distances. If the correlation between cost and Euclidean distance is quite high, it will be difficult to find an effect of forests on gene flow as the Euclidean distance does already explain most of the genetic variation between subpopulations.

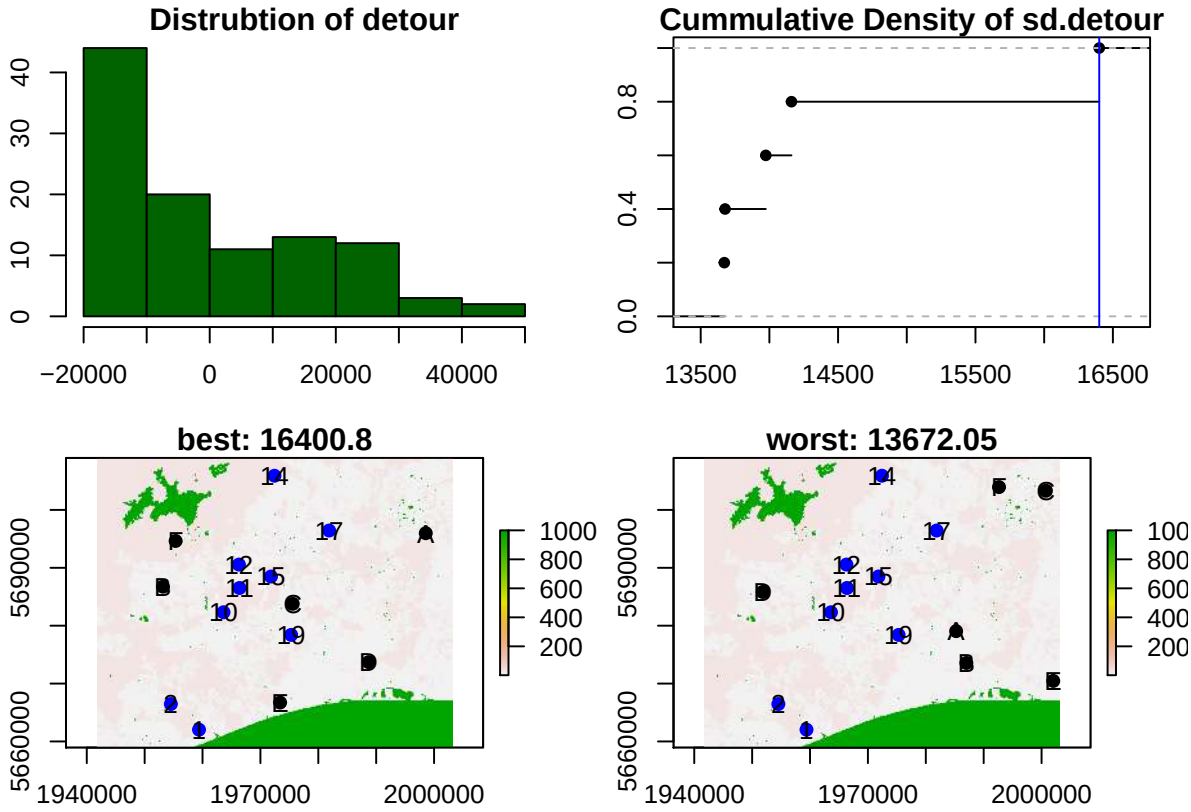
```
# calculate the correlation between the genetic, cost, and euclidian
# distances between populations
cor(cbind(gd, cd, ed))
```

```
##           gd        cd        ed
## gd 1.0000000 0.5364951 0.8260209
## cd 0.5364951 1.0000000 0.5887937
## ed 0.8260209 0.5887937 1.0000000
```

There is a significant relationship between the Euclidian distance and genetic distance but no correlation between the cost distance and the genetic distance. This could be the case because the cost and euclidian distances quite highly correlated (0.58).

While we were unable to find a significant effect of the landscape on the genetic distances between the nine sampled populations, from our post-hoc comparisons we can see that our sampling scheme was not ideal. If we would have additional funds and time to continue sampling, we could incorporate our already sampled possums as fixed populations and reoptimize the sampling design for more sampling locations.

```
fixed <- xys
opfix <- opt.landgen(landscape = forest, nlocations = 15, mindist = 5000, method = "leastcost",
    NN = 8, iter = 5, mask = mask, fixed = fixed)
```




By incorporating the additional 6 sampling locations into our analysis we could show the effect of forest on dispersal distances, something we could actually demonstrate using the full data set with 31 populations (see Sarre et a. 2014).

# References


Gruber, B. & Adamack, A.T. (2015). Landgenreport: a new R function to simplify landscape genetic analysis using resistance surface layers. Molecular Ecology Resources, 15, 1172 – 1178.

Sarre, S.D., Aitken, N., Adamack, A.T., Macdonald, A.J., Gruber, B. & Cowan, P. (2014). Creating new evolutionary pathways through bioinvasion: The population genetics of brushtail possums in New Zealand. Molecular Ecology, 23, 3419–3433.

Jombart, T. (2008). adegenet: a R package for the multivariate analysis of genetic markers. Bioinformatics, 24, 1403–1405.

van Etten, J. (2014). gdistance: distances and routes on geographical grids. R package version 1.1.5. http://cran.r-project.org/package=gdistance.