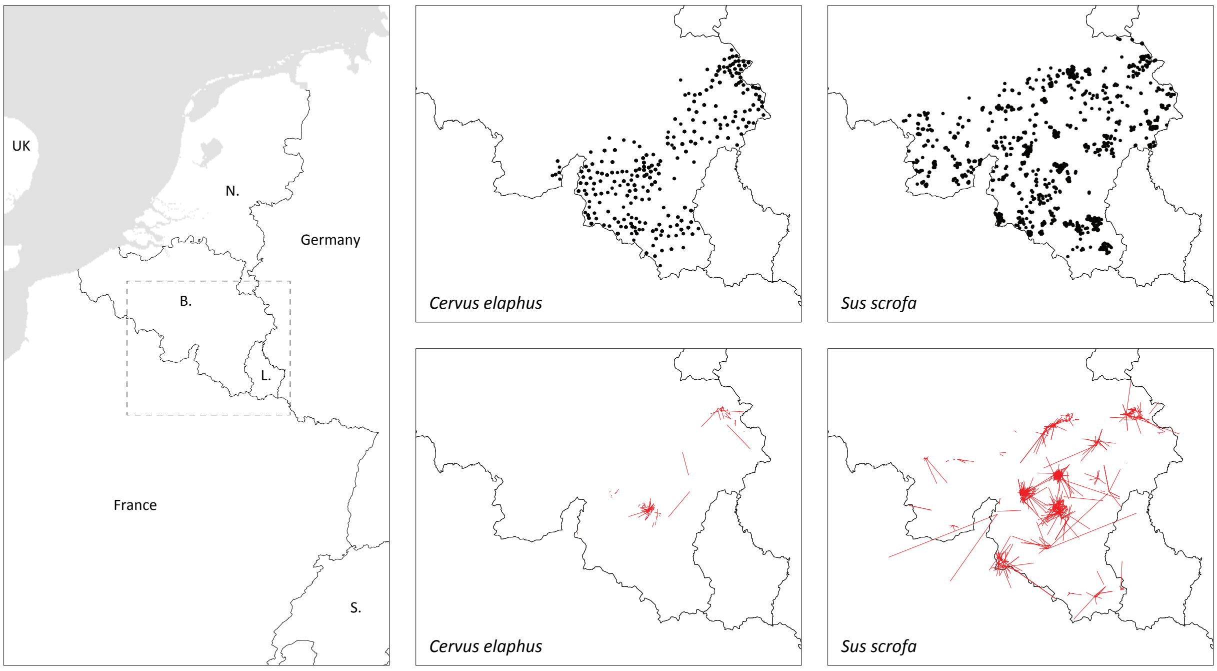


Figure S1 Maps of sampling locations (above) and maps displaying the CMR (capture-mark-recapture) movement vector data (below). On the global map (left), “N” refers to Netherlands, “B” to Belgium, “L” to Luxemburg, “S” to Switzerland and “UK” to United Kingdom.

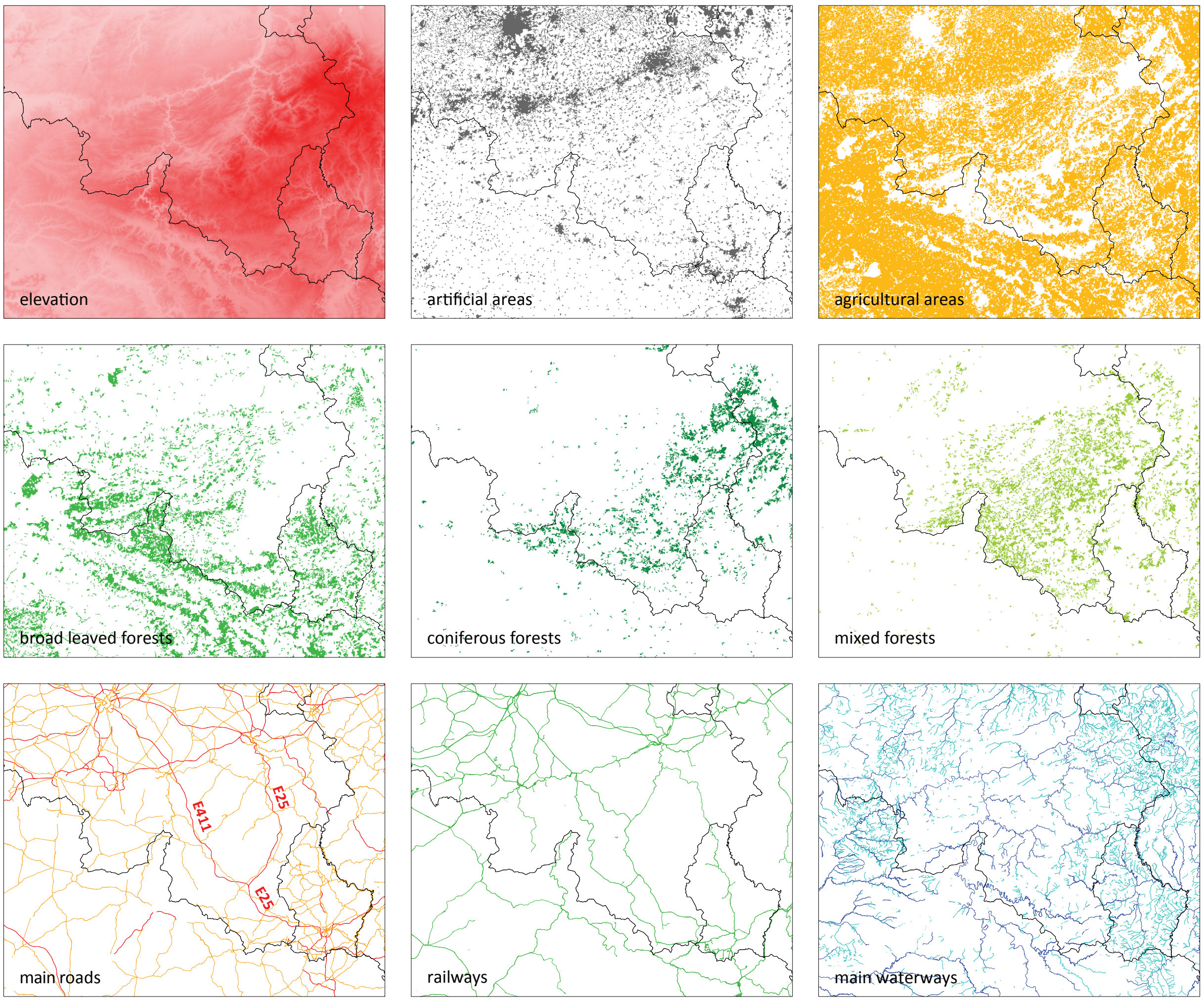


Figure S2 Environmental layers tested as potential factors influencing the genetic differentiation among *Cervus elaphus* and *Sus scrofa* individuals in Wallonia (raster cell size: 0.05 arcmin). On the “main roads” layer, red and orange lines respectively correspond to motorways and primary roads, and on the “main waterways” layer, blue and blue-green lines respectively correspond to rivers and streams.

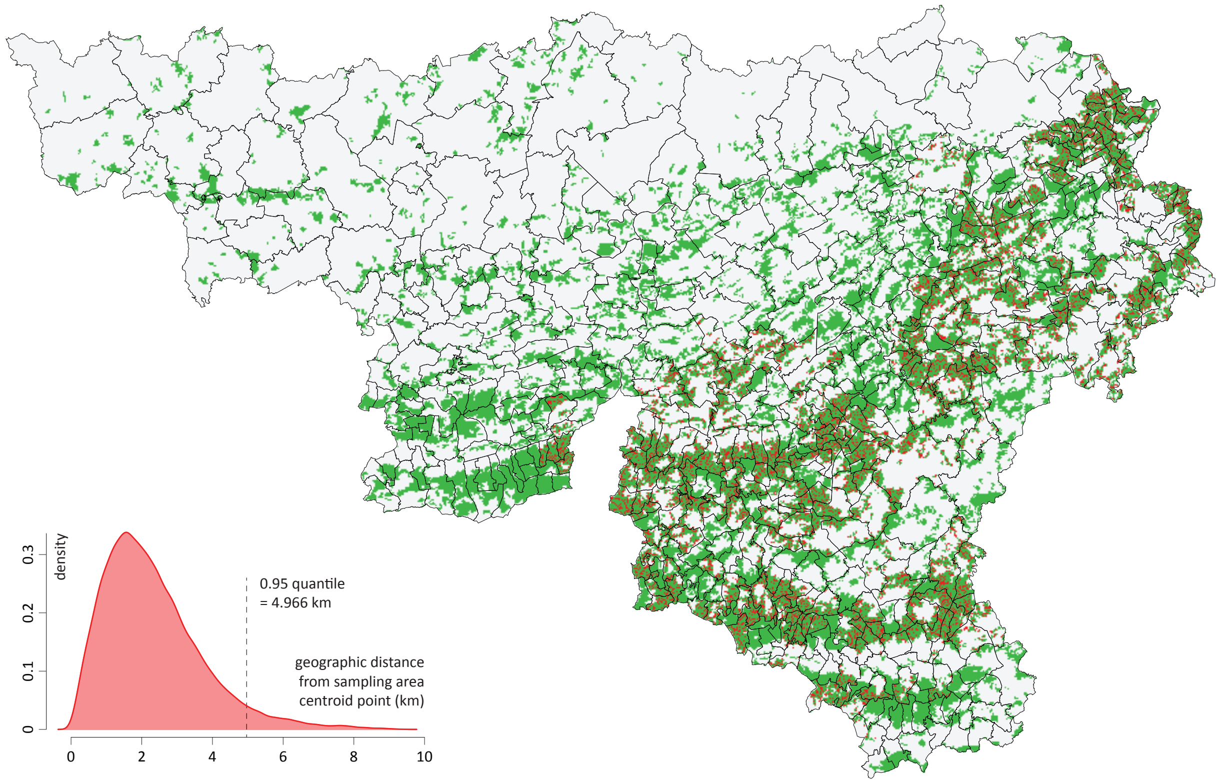


Figure S3. Map of Wallonia displaying the overall forest coverage (in green) and the hunting administrative areas (called *triages*) for *Cervus elaphus* (polygons with black contour). Contrary to *Sus scrofa* for which we have relatively precise sampling coordinates (see the text for further details), centroid points of these administrative areas correspond to the most precise sampling information available for *C. elaphus*. Red points indicate potential sampling locations randomly distributed within sampled administrative areas. The number of sampling locations simulated per administrative area is proportional to the area of the corresponding polygon (with a maximum of 10,000 locations simulated within the largest sampled administrative area). Simulated sampling locations falling outside the forest coverage have been eventually discarded. The remaining simulated sampling points have been used to estimate the distribution of geographic distances between potential sampling locations and centroid points of hunting administrative areas. The estimated distribution (on the left) was then used to determine a relevant radius that could realistically define the uncertainty related to the sampling precision of *C. elaphus*. Based on the 0.95 quantile of this distribution equal to 4.966 km, we define such a radius equal to 5 km for *C. elaphus.*

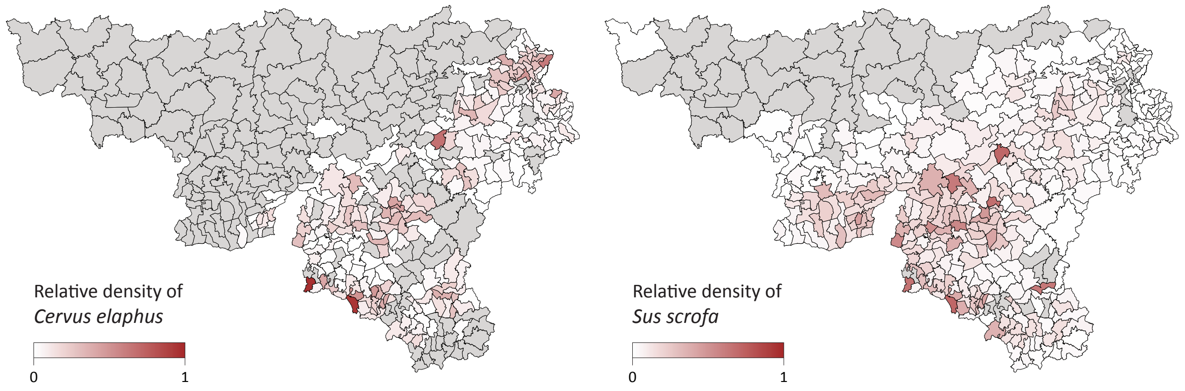


Fig. S4. Maps of relative density for *Cervus elaphus* and *Sus scrofa* in Wallonia. Relative densities were estimated per hunting administrative area (called *triages*) by dividing the total number of deaths (i.e. animals shouted and found deaths during the sampling period of each species) by the area of each polygon. The sampling period was 2003-2007 for *C. elaphus* and 2005-2013 for *S. scrofa*.

Figure S5 Summary of STRUCTURE results: log(P(*K*) and delta(*K*) against *K*.

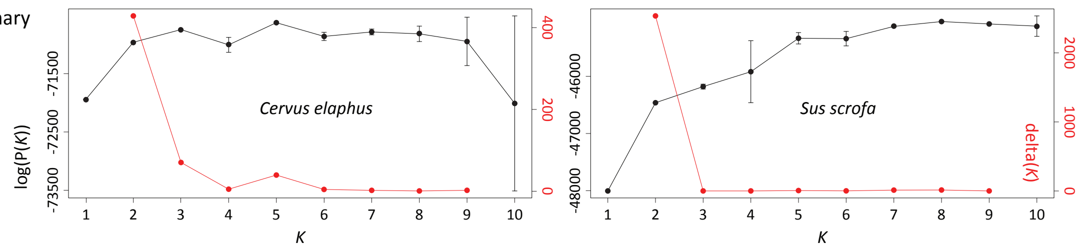
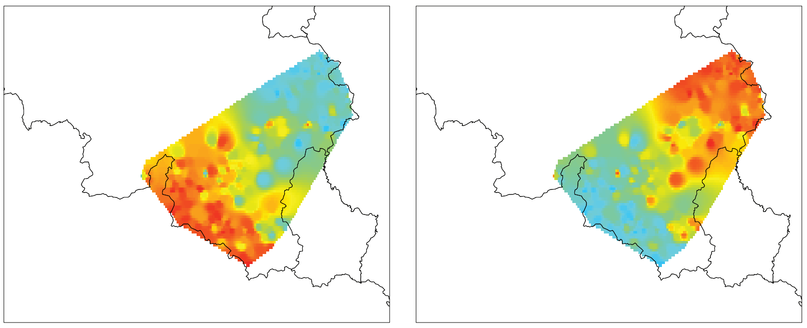


Figure S6 Interpolation graphs generated for the two clusters inferred by STRUCTURE (log(P(*K*)) method) for ***Cervus elaphus***.



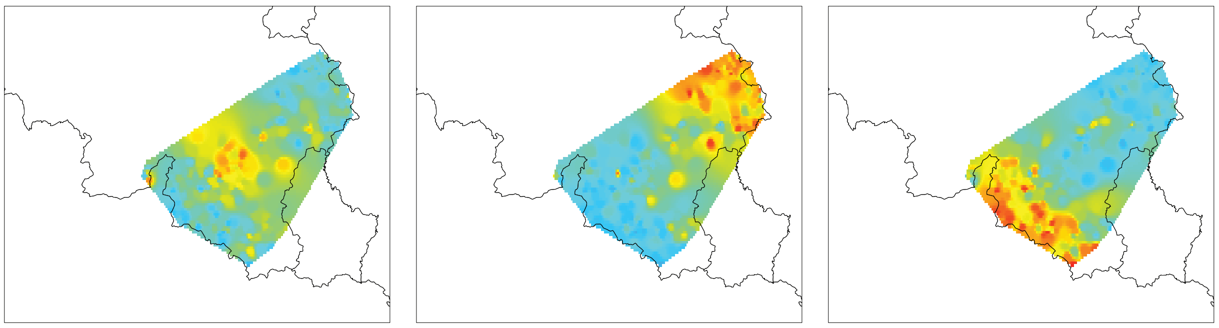


Figure S7 Interpolation graphs generated for the three clusters inferred by STRUCTURE (Evanno method) for ***Cervus elaphus***.

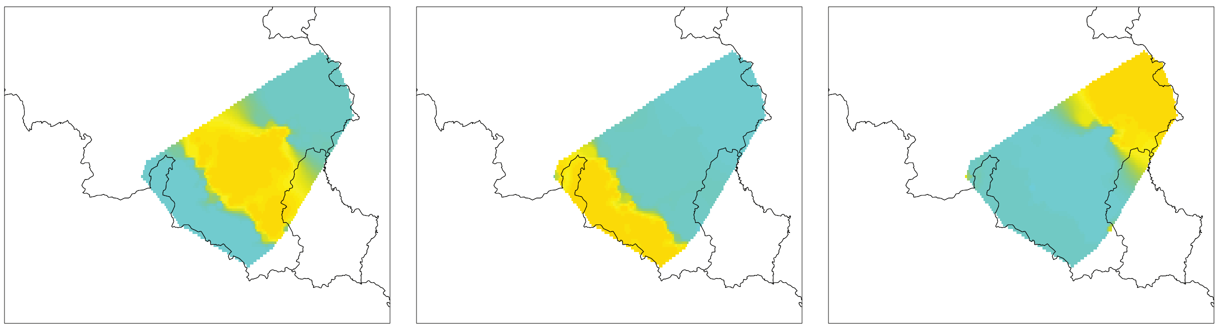


Figure S8 Interpolation graphs generated for the three clusters inferred by GENELAND for ***Cervus elaphus***.

Figure S9 Interpolation graphs generated for the two clusters inferred by STRUCTURE (log(P(*K*)) method) for ***Sus scrofa***.

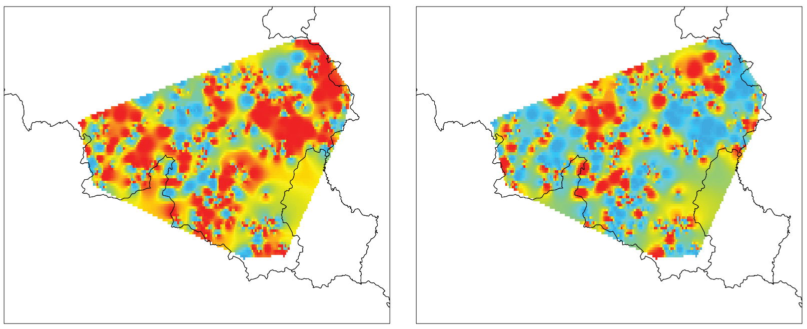


Figure S10 Interpolation graphs generated for the eight clusters inferred by STRUCTURE (Evanno method) for ***Sus scrofa***.

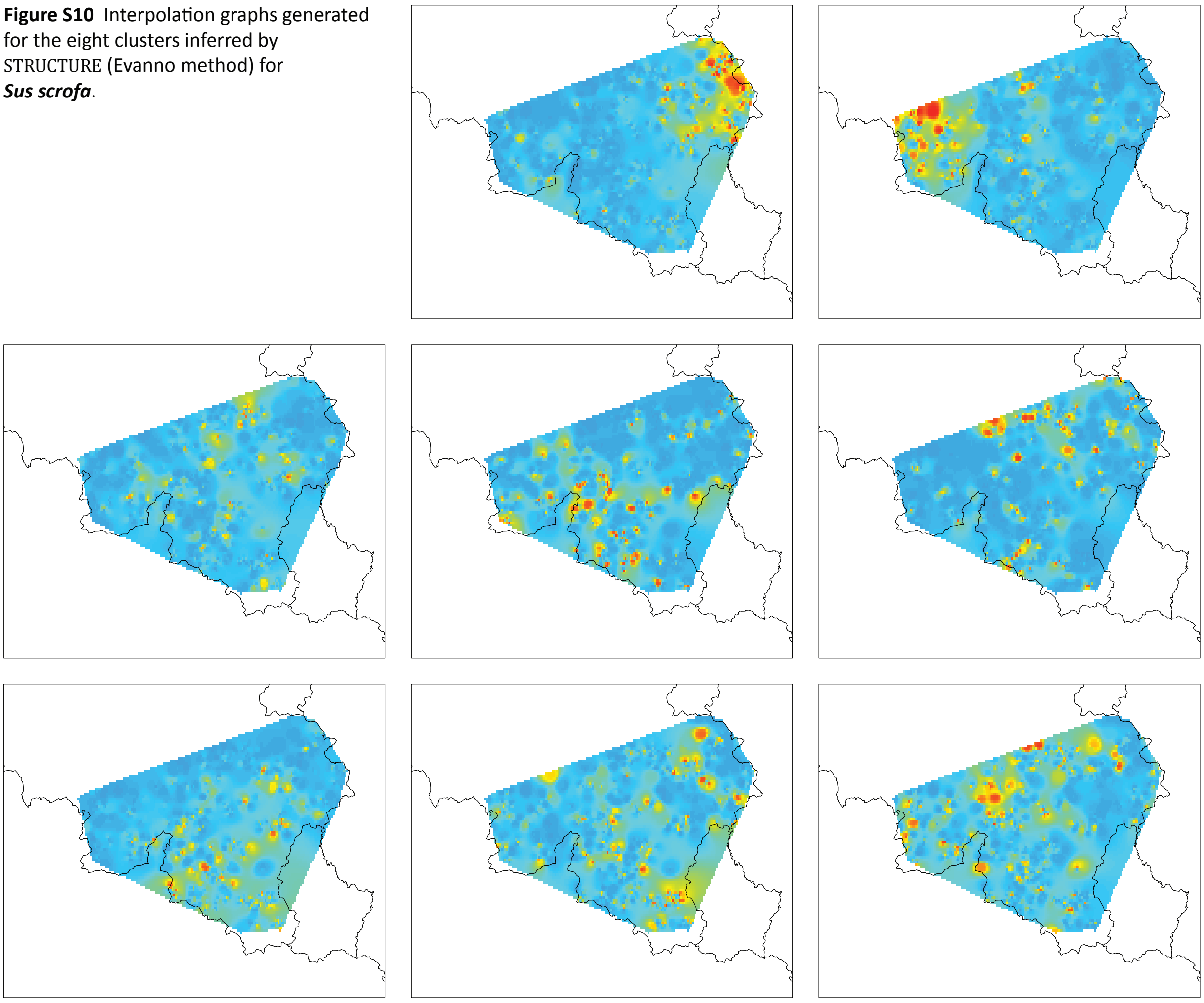
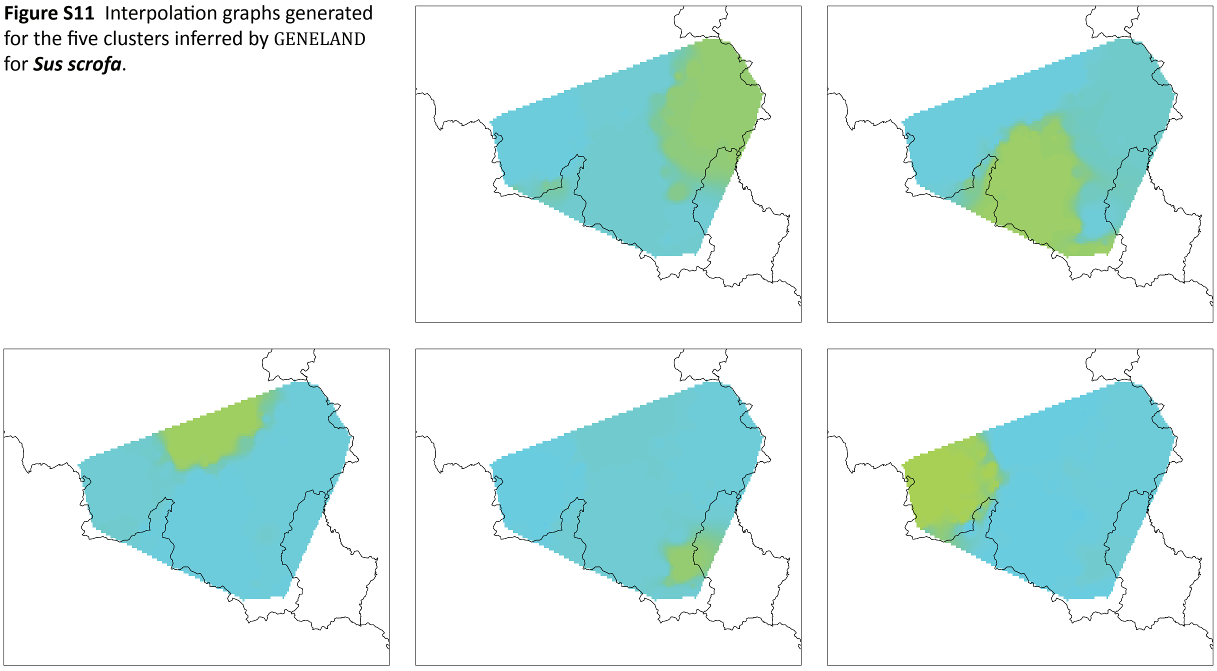


Figure S11 Interpolation graphs generated for the five clusters inferred by GENELAND for ***Sus scrofa***.



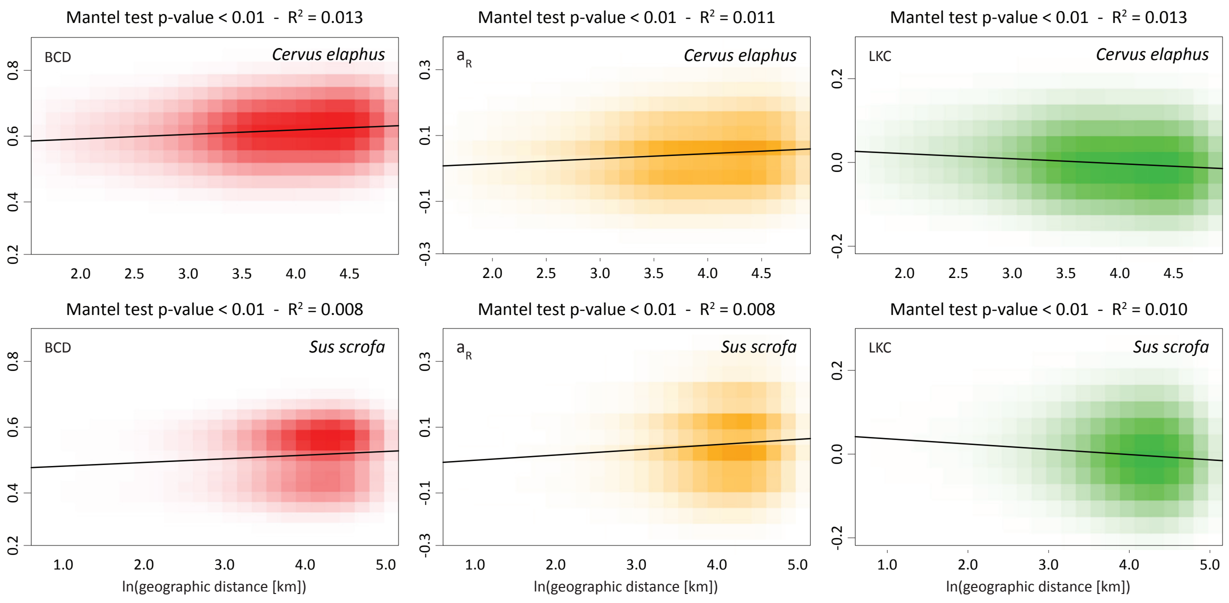


Figure S12 IBD (isolation-by-distance) analyses: density plots, Mantel tests and linear regressions performed with each genetic distance. N.B.: a negative slope is expected in the case of the LKC metric, which is a kinship coefficient measuring genetic similarity rather than dissimilarity.

Figure S13 Position and age of wildlife crossings in Wallonia (in green). Grey areas and red lines respectively correspond to artificial areas and motorways. Starting year of the different motorway segments are also reported on the map (in red; www.wegen-routes.be).

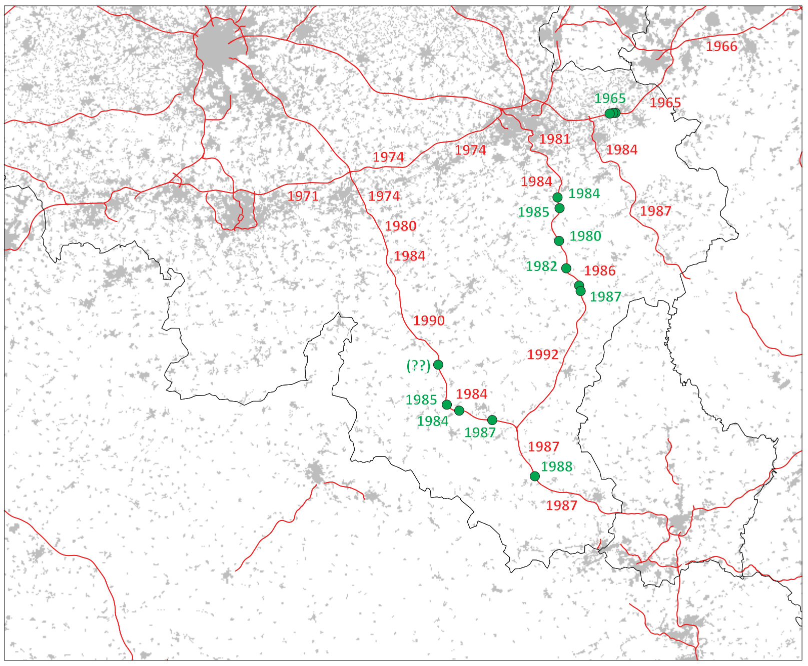


Table S1 Properties of the microsatellite loci used in this study. “N” refers to the number of samples successfully analysed, “A” to the number of alleles, “H_O” to the observed heterozygosity and “H_E” to the expected heterozygosity.

Cervus elaphus (red deer)						_Sus scrofa_ (wild boar)				
locus	**N**	**A**	**H_O**	**H_E**		**locus**	**N**	**A**	**H_O**	**H_E**
BM1818	1707	9	0.746	0.759		S0002	1230	16	0.774	0.800
Cer14	1704	15	0.695	0.739		S0005	1229	28	0.839	0.874
CSPS115	1704	14	0.803	0.811		S0026	1231	8	0.997	0.709
CSSM14	1702	4	0.088	0.087		S0090	1229	8	0.726	0.746
CSSM16	1707	8	0.576	0.605		S0097	1231	12	0.755	0.804
CSSM66	1706	12	0.658	0.752		S0155	1231	7	0.431	0.443
CSSM19	1707	13	0.775	0.808		S0226	1231	8	0.536	0.560
CSSM22	1706	6	0.660	0.661		Sw122	1231	7	0.539	0.613
ETH225	1704	14	0.827	0.863		Sw240	1231	14	0.713	0.701
Haut14	1706	14	0.858	0.875		Sw632	1231	9	0.598	0.591
ILSTS06	1695	14	0.799	0.840		Sw857	1230	8	0.667	0.727
INRA35	1706	10	0.762	0.781		Sw911	1231	6	0.527	0.526
MM12	1701	7	1.000	0.788		Sw936	1231	14	0.934	0.857
						Sw951	1231	5	0.022	0.022

Table S2 Results of the univariate analyses performed with the _pairwise approach_ - determination coefficient R^2 estimated from univariate regressions between genetic and environmental distances. (*) refers to significant R^2 values (p-value < 0.05), values in bold refer to R^2 value higher than R^2 value estimated for the null raster (in blue), italic values indicate that the associated coefficient beta of the linear regression is negative, and “C”/“R” indicate if the considered environmental raster was respectively treated as a conductance or resistance factor for the computation of environmental distances with circuit theory.

Environmental	_Cervus elaphus_ (red deer)			_Sus scrofa_ (wild boar)		
distances	**BCD**	**a_R**	**LKC**	**BCD**	**a_R**	**LKC**
Null raster (R)	0.015*	0.012*	_0.014_*	0.009*	0.009*	_0.010_*
Elevation (C)	0.012*	0.010*	_0.009_*	_0.001_*	0.001*	_0.003_*
Elevation (R)	0.004*	0.004*	_0.007_*	__0.014__*	0.004*	_0.002_*
Artificial areas (R)	0.003*	0.003*	_0.001_*	___0.013___*	_0.005_*	_0.000_*
Agricultural areas (R)	0.001*	0.000*	_0.003_*	0.001*	0.004*	_0.001_*
Broad leaved forests (C)	0.001*	0.002*	_0.002_*	0.000*	0.000*	_0.001_*
Coniferous forests (C)	0.005*	0.003*	_0.003_*	_0.000_*	0.001*	_0.002_*
Mixed forests (C)	0.000*	0.000*	_0.001_*	0.000	0.001*	_0.001_*
Motorways (R; _k_=10)	0.006*	0.005*	_0.008_*	0.001*	0.000*	_0.000_*
Motorways (R; _k_=100)	0.000*	0.000	_0.000_*	0.000*	0.000*	_0.000_*
Motorways (R; _k_=1,000)	_0.000_*	_0.000_*	_0.000_*	0.000*	0.001*	_0.001_*
Primary roads (R; _k_=10)	0.007*	0.006*	_0.005_*	0.003*	0.002*	_0.002_*
Primary roads (R; _k_=100)	0.002*	0.001*	_0.001_*	0.000*	0.000	_0.000_
Primary roads (R; _k_=1000)	0.001*	0.001*	_0.000_*	_0.000_*	0.000	_0.000_
Railways (R; _k_=10)	0.008*	0.006*	_0.008_*	_0.003_*	0.002*	_0.002_*
Railways (R; _k_=100)	0.001*	0.000*	_0.001_*	_0.000_	_0.000_*	_0.000_*
Railways (R; _k_=1000)	0.000*	_0.000_	_0.000_*	_0.000_	_0.000_*	_0.000_*
Rivers (R; _k_=10)	0.006*	0.006*	_0.004_*	0.003*	0.003*	_0.003_*
Rivers (R; _k_=100)	0.001*	0.001*	_0.000_*	_0.001_*	0.000*	_0.000_*
Rivers (R; _k_=1000)	0.000*	0.000*	_0.000_*	_0.000_*	0.000	_0.000_
Streams (R; _k_=10)	0.002*	0.003*	_0.002_*	_0.001_*	0.000*	_0.000_*
Streams (R; _k_=100)	0.000*	0.001*	_0.000_*	_0.000_	_0.000_*	_0.000_*
Streams (R; _k_=1000)	0.000	0.000*	_0.000_*	_0.000_*	_0.000_*	_0.000_*

Appendix S1

Using capture-markage-recapture data to study the impact of a barrier on dispersal frequency

Simon Dellicour

June 21, 2018

The present tutorial describes how to analyse capture-markage-recapture (CMR) data to study the impact of barriers on the dispersal frequency of individuals. In particular, this tutorial describes how analysing the impact of motorways on the dispersal frequency of wild boar (*Sus scrofa*) individuals in southern Belgium (Dellicour *et al. submitted*). The first step is to install and load the following R packages: “raster”, “rgdal”[1], “rgeos”, “sp” and “yhat”:

```
> install.packages("raster"); library(raster)
> install.packages("rgdal"); library(rgdal)
> install.packages("rgeos"); library(rrgeos)
> install.packages("sp"); library(sp)
> install.packages("yhat’"); library(yhat’)
```

This tutorial requires the CMR data example files for *S. scrofa* also available at `http://evolve.zoo.ox.ac.uk/Evolve/Software.html`. These files are “Sscrofa_CMR_data.csv”, a csv file containing the CMR records for *S. scrofa* (Dellicour *et al. submitted*), and “Motorways_shapefile”, a directory containing the motorways shapefile.

Step 1: plotting the motorways and CMR data

CMR data analysed in this tutorial are composed of a set of 1674 movement records for wild boar individuals (*S. scrofa*, Dellicour *et al. submitted*). Each CMR record is considered as an independent movement vector with a starting location, a dispersal duration and an nding location. The first step is here to plot the different data, i.e. the CMR movement vectors as well as the motorways on the study area. The CMR data, motorways shapefile and template raster can be loaded from the working directory with the following commands:

```
> motorways_shapefile = readOGR(dsn="./Motorways_shapefile",
   layer="Motorways_shapefile")
> tab = read.csv("Sscrofa_CMR_data.csv", header=T, sep=",")
```

We can then plot the different elements as follows:

```
> plot(template_raster, col="white", legend=F)
> plot(motorways_shapefile, col="red", add=T)
> segments(tab[,"x0"], tab[,"y0"], tab[,"x1"], tab[,"y1"], col="black", lwd=0.5)
```

[1]The “rgdal” package requires the preliminary installation of GDAL (Geospatial Data Abstraction Library), a C++ library for reading and writing raster geospatial data formats. See `http://www.gdal.org` for further details.

The resulting graph is displayed in Figure 1.

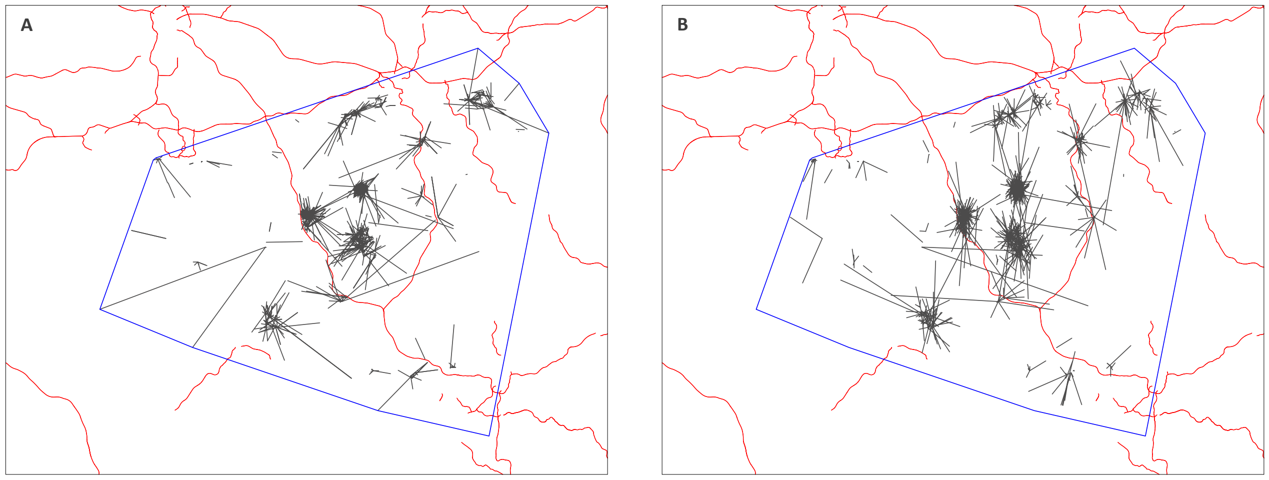


Figure 1: (**A**) CMR (capture-mark-recapture) records available for wild boar (*Sus scrofa*) individuals in southern Belgium (Wallonia). Red and black lines respectively indicate the position motorways and CMR movement vectors. (**B**) Example of CMR movement randomisation within a minimum convex hull defined by initial vector positions (blue contour).

Step 2: creating one SpatialLinesDataFrame per motorway

The second step is to create a distinct "SpatialLinesDataFrame" object per motorway segment and to store these new object in a "motorways_list":

```
> motorways_names = motorways_shapefile@data[,"ref"]
> motorways_list = list()
> for (i in 1:length(unique(motorways_names)))
>    {
>       ids = which(motorways_names == unique(motorways_names)[i])
>       motorway = motorways_shapefile
>       motorway@lines = motorway@lines[ids]
>       motorway@data = motorway@data[ids,]
>       motorways_list[[i]] = motorway
>    }
```

Step 3: computing the branch distances and the minimum distance between each branch and motorway

The third step is to compute the minimum distance between each branch and the nearest motorway segment. This step is necessary to avoid testing the impact of motorways on CMR movement vectors that could not have crossed such barrier anyway.

```
> branch_distances = matrix(nrow=dim(tab)[1], ncol=1)
> minimum_distances = matrix(nrow=dim(tab)[1], ncol=length(motorways_list))
> for (i in 1:dim(tab)[1])
>    {
>       pt1 = SpatialPoints(cbind(tab[i,"x0"],tab[i,"y0"]))
>       pt2 = SpatialPoints(cbind(tab[i,"x1"],tab[i,"y1"]))
>       branch_distances[i,1] = gDistance(pt1, pt2)
>       for (j in 1:length(motorways_list))
>          {
>             minimum_distances[i,j] = gDistance(pt1, motorways_list[[j]])
>          }
>    }
```

Step 4: counting the number of observed crossing motorway events

The fourth step is to count the number of times CMR movement vectors cross motorway segments (No). It is important to note that we will here consider that a CMR movement vector actually crosses a motorway segment if the number of intersections with this segment is an odd number. Indeed, an even number of intersections does not guarantee that the individual actually crossed the motorway. Furthermore, as the motorways are by nature composed of two distinct lines (one line per traffic direction), we also have to divide the numbers of identified intersections per two to assess if they correspond to odd or even numbers.

```
> No = 0
> for (i in 1:dim(tab)[1])   {
>       nS = 0
>       ids = which(minimum_distances[i,]<branch_distances[i,1])
>       if (length(ids) > 0)
>          {
>             x = c(tab[i,"x0"],tab[i,"x1"])
>             y = c(tab[i,"y0"],tab[i,"y1"])
>             branch = SpatialLines(list(Lines(Line(cbind(x,y)), ID=i)))
>             for (j in 1:length(ids)) {
>                   n = 0
>                   intersections = gIntersection(motorways_list[[ids[j]]], branch)
>                   if (!is.null(intersections))
>                      {
>                         if (odd(dim(intersections@coords)[1]/2)) n = n + 1
>                         points(intersections, col="green3")
>                      }
>                   nS = nS + n
>                }
>          }
>       No = No + nS
>    }
> print(No)

22
```

Step 5: creating a minimum convex hull around CMR records

The fifth step is to create a minimum convex hull around CMR records:

```
> points1 = tab[,c("x0","y0")]; points2 = tab[,c("x1","y1")]
> colnames(points1) = c("x","y"); colnames(points2) = c("x","y")
> points = rbind(points1, points2)
> hull = chull(points)
> hull = c(hull,hull[1])
> hull = Polygon(points[hull,])
> hull = Polygons(list(hull),1)
> hull = SpatialPolygons(list(hull))
> hull_raster = mask(template_raster, hull, snap="out")
> plot(hull, add=T, border="blue")
```

As coded above, the minimum convex hull is subsequently used to create a “hull_raster” object that will be useful to constraint the CMR randomisations performed in step 6.

Step 6: randomisation step for testing the significance level of No

The last step is to use a randomisation procedure to assess the level of significance of the observed number of crossing motorway events (No). The randomisation procedure requires the preliminary definition of a “rotation” function:

```
> rotation = function(pt1, pt2, angle)
>    {
>       s = sin(angle); c = cos(angle)
>       x = pt2[1]-pt1[1]; y = pt2[2]-pt1[2]
>       x_new = (x*c)-(y*s); y_new = (x*s)+(y*c)
>       x_new = x_new+pt1[1]; y_new = y_new+pt1[2]
>       return(c(x_new,y_new))
>    }
```

This function basically allows to randomly rotate each CMR movement vector around its starting point and while maintaining the whole vector within the minimum convex hull created in step 5. The following randomisation procedure thus uses the "rotation" fonction to randomise all the CMR movement vectors to compute and generate a null distribution of numbers of crossing motorway events (Fig. 1). No is eventually compared to this null distribution to estimate a p-value reflecting its level of significance:

```
> nberOfRandomisations = 1000
> Ns = matrix(nrow=nberOfRandomisations, ncol=1)
> for (s in 1:nberOfRandomisations)
>    {
>       N = 0
>       for (i in 1:dim(tab)[1])
>          {
>             nS = 0
>             ids = which(minimum_distances[i,]<branch_distances[i,1])
>             if (length(ids) > 0)
>                {
>                   pt1 = c(tab[i,"x0"],tab[i,"y0"])
>                   pt2 = c(tab[i,"x1"],tab[i,"y1"])
>                   rotationWithinHull = FALSE
>                   while (rotationWithinHull == FALSE)
>                      {
>                         angle = (2*pi)*runif(1)
>                         pt2_rotated = rotation(pt1, pt2, angle)
>                         if (!is.na(extract(hull_raster, cbind(pt2_rotated[1],pt2_rotated[2]))))
>                            {
>                               rotationWithinHull = TRUE
>                            }
>                      }
>                   x = c(pt1[1],pt2_rotated[1])
>                   y = c(pt1[2],pt2_rotated[2])
>                   branch = SpatialLines(list(Lines(Line(cbind(x,y)), ID=i)))
>                   for (j in 1:length(ids)) {
>                         n = 0
>                         intersections = gIntersection(motorways_list[[ids[j]]], branch)
>                         if (!is.null(intersections))
>                            {
>                               if (odd(dim(intersections@coords)[1]/2)) n = n + 1
>                            }
>                         nS = nS + n
>                      }
>                }
>             N = N + nS
>          }
>       print(N)
>       Ns[s,1] = N
>    }
> write.table(Ns, "Ns.txt", col.names=F, row.names=F)
> pValue = sum(Ns<=No)/length(Ns); print(pValue)

0
```