A tutorial for the spatial Analysis of Principal Components (sPCA) using *adegenet* 2.1.5

Thibaut Jombart *

Imperial College London

MRC Centre for Outbreak Analysis and Modelling

October 6, 2021

Abstract

This vignette provides a tutorial for the spatial analysis of principal components (sPCA, [1]) using the *adegenet* package [2] for the R software [3]. sPCA is first illustrated using a simple simulated dataset, and then using empirical data of Chamois (*Rupicapra rupicapra*) from the Bauges mountains (France). In particular, we illustrate how sPCA complements classical PCA by being more powerful for retrieving non-trivial spatial genetic patterns.

*tjombart@imperial.ac.uk

Contents

1 Introduction **3**
1.1 Rationale of sPCA 3
1.2 The `spca` function 4
1.3 Contents of a `spca` object 9
1.4 Graphical display of `spca` results 13

2 Case study: spatial genetic structure of the chamois in the Bauges mountains **25**
2.1 An overview of the data 26
2.2 Summarising the genetic diversity 28
2.3 Mapping and testing PCA results 34
2.4 Multivariate tests of spatial structure 41
2.5 Spatial Principal Component Analysis 42

1 Introduction

This tutorial goes through the *spatial Principal Component Analysis* (sPCA, [1]), a multivariate method devoted to the identification of spatial genetic patterns. The purpose of this tutorial is to provide guidelines for the application of sPCA as well as to illustrate its usefulness for the investigation of spatial genetic patterns. After briefly going through the rationale of the method, we introduce the different tools implemented for sPCA in *adegenet*. This technical overview is then followed by the analysis of an empirical dataset which illustrates the advantage of sPCA over classical PCA for investigating spatial patterns.

1.1 Rationale of sPCA

Mathematical notations used in this tutorial are identical to the original publication [1]. The sPCA analyses a matrix of relative allele frequencies $\mathbf{X}$ which contains genotypes or populations (later refered to as 'entities') in rows and alleles in columns. Spatial information is stored inside a spatial weighting matrix $\mathbf{L}$ which contains positive terms corresponding to some measurement (often binary) of spatial proximity among entities. Most often, these terms can be derived from a connection network built upon a given algorithm (for instance, pp.752-756 in [4]). This matrix is row-standardized (*i.e.*, each of its rows sums to one), and all its diagonal terms are zero. $\mathbf{L}$ can be used to compute the spatial autocorrelation of a given centred variable $\mathbf{x}$ (`i.e.`, with mean zero) with n observations ($\mathbf{x} \in \mathbb{R}^n$) using Moran's I [5, 6, 7]:

$$I(\mathbf{x}) = \frac{\mathbf{x}^T\mathbf{L}\mathbf{x}}{\mathbf{x}^T\mathbf{x}} \tag{1}$$

In the case of genetic data, $\mathbf{x}$ contains frequencies of an allele. Moran's I can be used to measure spatial structure in the values of $\mathbf{x}$: it is highly positive when values of $\mathbf{x}$ observed at neighbouring sites tend to be similar (positive spatial autocorrelation, referred to as *global structures*), while it is strongly negative when values of $\mathbf{x}$ observed at neighbouring sites tend to be dissimilar (negative spatial autocorrelation, referred to as *local structures*).

However, since it is standardized by the variance of $\mathbf{x}$, Moran's index measures only spatial structures and not genetic variability. The sPCA defines the following function to measure both spatial structure and variability in $\mathbf{x}$:

$$C(\mathbf{x}) = \mathrm{var}(\mathbf{x})I(\mathbf{x}) = \frac{1}{n}\mathbf{x}^T\mathbf{L}\mathbf{x} \tag{2}$$

$C(\mathbf{x})$ is highly positive when $\mathbf{x}$ has a large variance and exhibits a global structure; conversely, it is largely negative when $\mathbf{x}$ has a high variance and displays a local structure. This function is the criterion used in sPCA, which finds linear combinations of the alleles of $\mathbf{X}$ (denoted $\mathbf{X}\mathbf{v}$) decomposing C from its maximum to its minimum value. Because $C(\mathbf{X}\mathbf{v})$ is a product of variance and autocorrelation, it is important, when interpreting the results, to detail both components and to compare their value with their range of variation (maximum attainable variance, as well as maximum and minimum I are known analytically). A structure with a low spatial autocorrelation can barely be interpreted as a spatial pattern; similarly, a structure with a low variance would likely not reflect any genetic structure. We will later see how these information can be retrieved from `spca` results.

1.2 The spca function

The simulated dataset used to illustrate this section has been analyzed in [1], and corresponds to Figure 2A of the article. In *adegenet*, the matrix of alleles frequencies previously denoted **X** exactly corresponds to the `@tab` slot of `genind` or `genpop` objects:

```
library(adegenet)
```

```
library(spdep)
data(spcaIllus)
obj <- spcaIllus$dat2A
obj

## /// GENIND OBJECT /////////
##
##  // 80 individuals; 20 loci; 192 alleles; size: 116.2 Kb
##
##  // Basic content
##    @tab:  80 x 192 matrix of allele counts
##    @loc.n.all: number of alleles per locus (range: 6-14)
##    @loc.fac: locus factor for the 192 columns of @tab
##    @all.names: list of allele names for each locus
##    @ploidy: ploidy of each individual  (range: 2-2)
##    @type:  codom
##    @call: old2new(object = obj)
##
##  // Optional content
##    @pop: population of each individual (group size range: 10-35)
##    @other: a list containing: xy

head(tab(obj[loc = 1]))

##      L01.1 L01.2 L01.3 L01.4 L01.5 L01.6 L01.7 L01.8 L01.9
## 0035     0     0     0     0     1     1     0     0     0
## 0352     0     0     1     0     1     0     0     0     0
## 0423     0     0     0     0     1     0     0     0     1
## 0289     0     0     0     0     0     1     0     0     1
## 0487     0     0     0     0     0     1     0     1     0
## 0053     0     0     0     0     1     1     0     0     0
```

The object `obj` is a `genind` object; note that here, we only displayed the table for the first locus (`loc=1`).

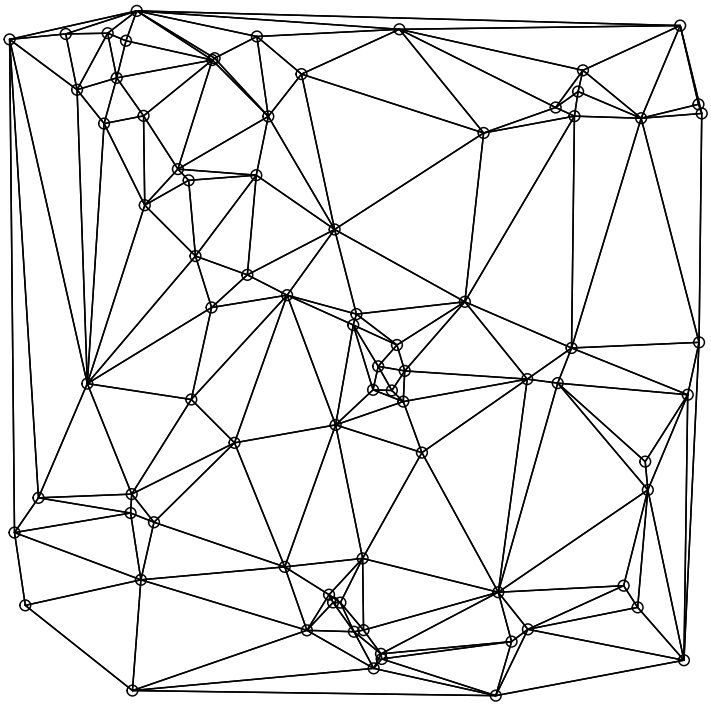
The function performing the sPCA is `spca`. As of *adegenet* 2.1.0, it is a generic with methods for various types of objects. The `genind` method accepts a bunch of arguments, but

only the first two are mandatory to perform the analysis (see `?spca` for further information):

```
methods("spca")

## [1] spca.data.frame spca.default    spca.genind     spca.genpop     spca.matrix
## see '?methods' for accessing help and source code

args(spca.genind)

## function (obj, xy = NULL, cn = NULL, matWeight = NULL, scale = FALSE,
##     scannf = TRUE, nfposi = 1, nfnega = 1, type = NULL, ask = TRUE,
##     plot.nb = TRUE, edit.nb = FALSE, truenames = TRUE, d1 = NULL,
##     d2 = NULL, k = NULL, a = NULL, dmin = NULL, ...)
## NULL
```

The argument `obj` is a `genind` object. By definition in sPCA, the studied entities are georeferenced. The spatial information can be provided to the function `spca` in several ways, the first being through the `xy` argument, which is a matrix of spatial coordinates with 'x' and 'y' coordinates in columns. Alternatively, these coordinates can be stored inside the `genind`/`genpop` object, preferably as `@other$xy`, in which case the `spca` function will detect and use this information, and not request an `xy` argument. Note that `obj` already contains spatial coordinates at the appropriate place. Hence, we can use the following command to run the sPCA (`ask` and `scannf` are set to FALSE to avoid interactivity):

```
mySpca <- spca(obj, ask=FALSE, type=1, scannf=FALSE)
```

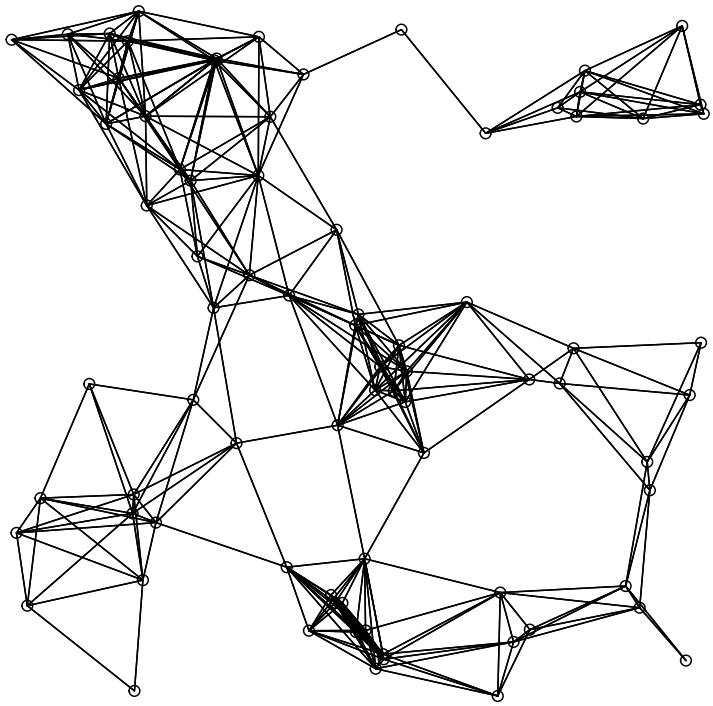


Note, however, that spatial coordinates are not directly used in sPCA: the spatial information is included in the analysis by the spatial weighting matrix $\mathbf{L}$ derived from a connection network (eq. 1 and 2). Technically, the `spca` function can incorporate spatial weightings as a matrix (argument `matWeight`), as a connection network with the classes `nb` or `listw` (argument `cn`), both implemented in the `spdep` package. The function `chooseCN` is a wrapper for different functions scattered across several packages implementing a variety of connection networks. If only spatial coordinates are provided to `spca`, `chooseCN` is called to construct an appropriate graph. See `?chooseCN` for more information. Note that many of the `spca` arguments are in fact arguments for `chooseCN`: `type`, `ask`, `plot.nb`, `edit.nb`, `d1`, `d2`, `k`, `a`, and `dmin`. For instance, the command:

```
mySpca <- spca(obj,type=1,ask=FALSE,scannf=FALSE)
```

performs a sPCA using the Delaunay triangulation as connection network (`type=1`, see `?chooseCN`), while the command:

```
mySpca <- spca(obj,type=5,d1=0,d2=2,scannf=FALSE)
```



computes a sPCA using a connection network which defines neighbouring entities based on pairwise geographic distances (`type=5`), considering as neighbours two entities whose distance between 0 (`d1=0`) and 2 (`d2=2`).

Another possibility is of course to provide directly a connection network (`nb` object) or a list of spatial weights (`listw` object) to the `spca` function; this can be done via the `cn` argument. For instance:

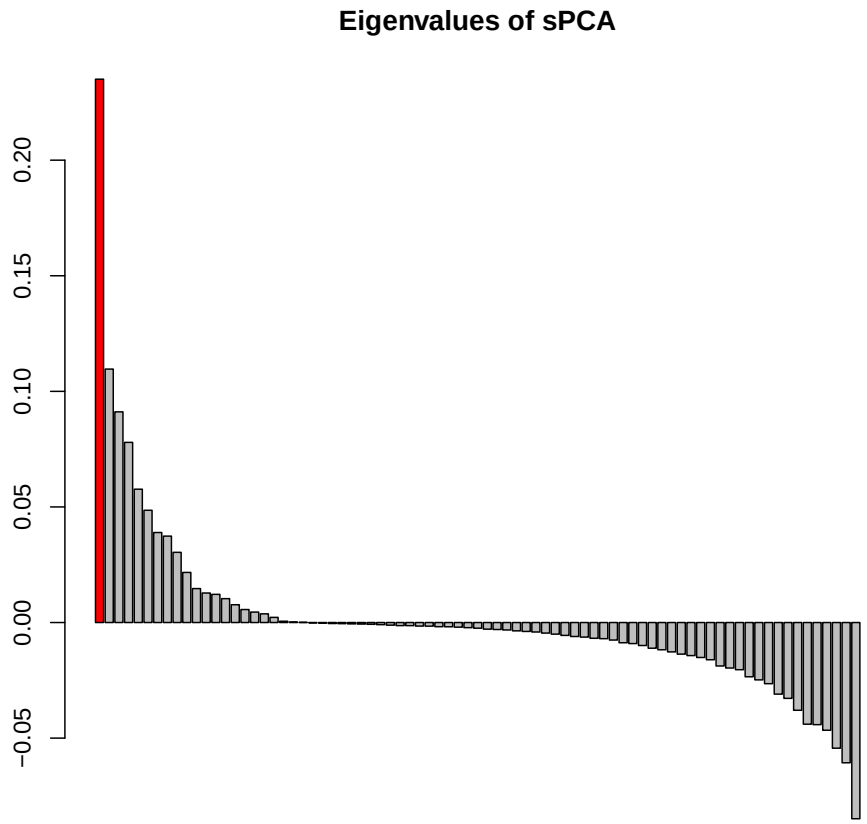
```
myCn <- chooseCN(obj$other$xy, type=6, k=10, plot=FALSE)
myCn

## Neighbour list object:
## Number of regions: 80
## Number of nonzero links: 932
## Percentage nonzero weights: 14.5625
## Average number of links: 11.65
```

```
class(myCn)

## [1] "nb"

mySpca2 <- spca(obj,cn=myCn,scannf=FALSE)
```

produces a sPCA using `myCn` ($k = 10$ nearest neighbours) as a connection network.

When used interactively (`scannf=TRUE`), `spca` displays a barplot of eigenvalues and asks the user for a number of positive axes ('first number of axes') and negative axes ('second number of axes') to be retained. For the object `mySpca`, this barplot would be (here we indicate in red the retained eigenvalue):

```
barplot(mySpca$eig,main="Eigenvalues of sPCA", col=rep(c("red","grey"),c(1,100)))
```



Positive eigenvalues (on the left) correspond to global structures, while negative eigenvalues (on the right) indicate local patterns. Actual structures should result in more extreme (positive or negative) eigenvalues; for instance, the object `mySpca` likely contains one single global structure, and no local structure. If one does not want to choose the number of

retained axes interactively, the arguments `nfposi` (number of retained factors with positive eigenvalues) and `nfnega` (number of retained factors with negative eigenvalues) can be used. Once this information has been provided to `spca`, the analysis is computed and stored inside an object with the class `spca`.

1.3 Contents of a spca object

Let us consider a `spca` object resulting from the analysis of the object `obj`, using a Delaunay triangulation (`type=1`) as connection network:

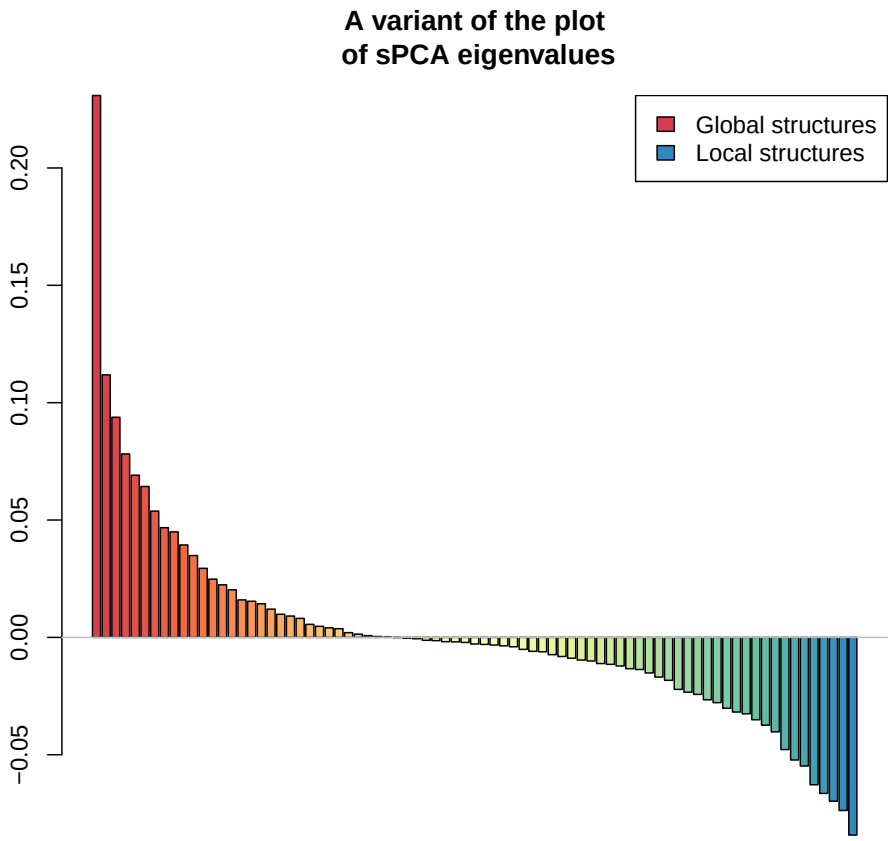
```
mySpca <- spca(obj,type=1,scannf=FALSE,plot.nb=FALSE,nfposi=1,nfnega=0)
class(mySpca)

## [1] "spca"

mySpca

##  ########################################
##  # spatial Principal Component Analysis #
##  ########################################
## class: spca
## $call: spca.genind(obj = obj, scannf = FALSE, nfposi = 1, nfnega = 0,
##     type = 1, plot.nb = FALSE)
##
## $nfposi: 1 axis-components saved
## $nfnega: 0 axis-components saved
## Positive eigenvalues: 0.2309 0.1118 0.09379 0.07817 0.06911 ...
## Negative eigenvalues: -0.08421 -0.07376 -0.06978 -0.06648 -0.06279 ...
##
##   vector length mode    content
## 1 $eig   79     numeric eigenvalues
##
##   data.frame nrow ncol content
## 1 $tab       80   192  transformed data: optionally centred / scaled
## 2 $c1        192  1    principal axes: scaled vectors of alleles loadings
## 3 $li        80   1    principal components: coordinates of entities ('scores')
## 4 $ls        80   1    lag vector of principal components
## 5 $as        2    1    pca axes onto spca axes
##
## $xy: matrix of spatial coordinates
## $lw: a list of spatial weights (class 'listw')
##
## other elements: lw
```

An `spca` object is a list containing all required information about a performed sPCA. Details about the different components of such a list can be found in the `spca` documentation (`?spca`).

The purpose of this section is to explicit how the elements described in [1] are stored inside a `spca` object.

First, eigenvalues of the analysis are stored inside the `$eig` component as a numeric vector stored in decreasing order:

```
head(mySpca$eig)

## [1] 0.23087862 0.11184721 0.09378750 0.07816561 0.06910536 0.06429596

tail(mySpca$eig)

## [1] -0.05480010 -0.06279067 -0.06647896 -0.06978457 -0.07375563 -0.08421213

length(mySpca$eig)

## [1] 79

barplot(mySpca$eig, main="A variant of the plot\n of sPCA eigenvalues",
        col=spectral(length(mySpca$eig)))
legend("topright", fill=spectral(2),
       leg=c("Global structures", "Local structures"))
abline(h=0,col="grey")
```



The axes of the analysis, denoted **v** in eq. (4) [1] are stored as columns inside the `$c1` component. Each column contains loadings for all the alleles:

```
head(mySpca$c1)

##              Axis 1
## L01.1  1.268838e-02
## L01.2  5.551115e-17
## L01.3 -1.119979e-01
## L01.4 -2.220446e-16
## L01.5 -2.766095e-02
## L01.6 -4.477031e-02

tail(mySpca$c1)

##            Axis 1
## L20.3  0.28715850
## L20.4  0.01485180
## L20.5 -0.01500353
```

```
## L20.6  0.01659481
## L20.7 -0.14260743
## L20.8 -0.15388988

dim(mySpca$c1)

## [1] 192   1
```

The entity scores, denoted $\psi = \mathbf{Xv}$ in the article, are stored in columns in the `$li` component:

```
head(mySpca$li)

##          Axis 1
## 0035 -0.4367748
## 0352 -0.8052723
## 0423 -0.4337114
## 0289  0.1434650
## 0487 -0.4802931
## 0053 -0.5421831

tail(mySpca$li)

##           Axis 1
## 1074 -0.06178196
## 1187 -0.08144162
## 1260  0.41491795
## 1038  0.25643986
## 1434  0.35618737
## 1218  0.21433977

dim(mySpca$li)

## [1] 80  1
```

The lag vectors of the scores can be used to better perceive global structures. Lag vectors are stored in the `$ls` component:

```
head(mySpca$ls)

##          Axis 1
## 0035 -0.7076732
## 0352 -0.6321654
## 0423 -0.4822952
## 0289  0.3947791
## 0487 -0.2803381
```

```
## 0053 -0.4848376
```

```
tail(mySpca$ls)
```

```
##          Axis 1
## 1074  0.4930238
## 1187 -0.8384871
## 1260  0.6887072
## 1038  0.3665794
## 1434  0.3109197
## 1218  0.3329688
```

```
dim(mySpca$ls)
```

```
## [1] 80  1
```

Lastly, we can compare the axes of an classical PCA (denoted **u** in the paper) to the axes of the sPCA (**v**). This is achieved by projecting **u** onto **v**, but this projection is a particular one: because both **u** and **v** are centred to mean zero and scaled to unit variance, the value of the projection simply is the correlation between both axes. This information is stored inside the `$as` component:

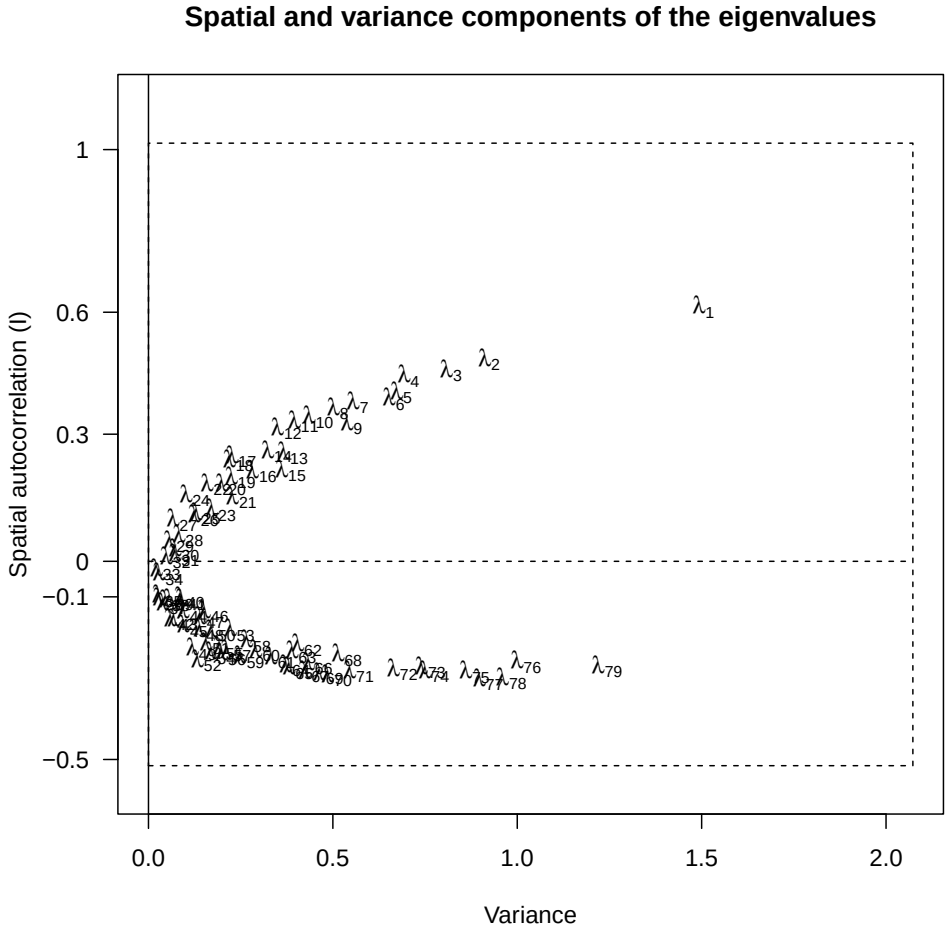
```
mySpca$as
```

```
##               Axis 1
## PCA Axis1 -0.7363595
## PCA Axis2  0.3395674
```

1.4 Graphical display of `spca` results

The information contained inside a `spca` object can be displayed in several ways. While we have seen that a simple barplot of sPCA eigenvalues can give a first idea of the global and local structures to be retained, we have also seen that each eigenvalue can be decomposed into a *variance* and a *spatial autocorrelation* (Moran's I) component. This information is provided by the `summary` function, but it can also be represented graphically. The corresponding function is `screeplot`, and can be used on any `spca` object:

```
screeplot(mySpca)
```

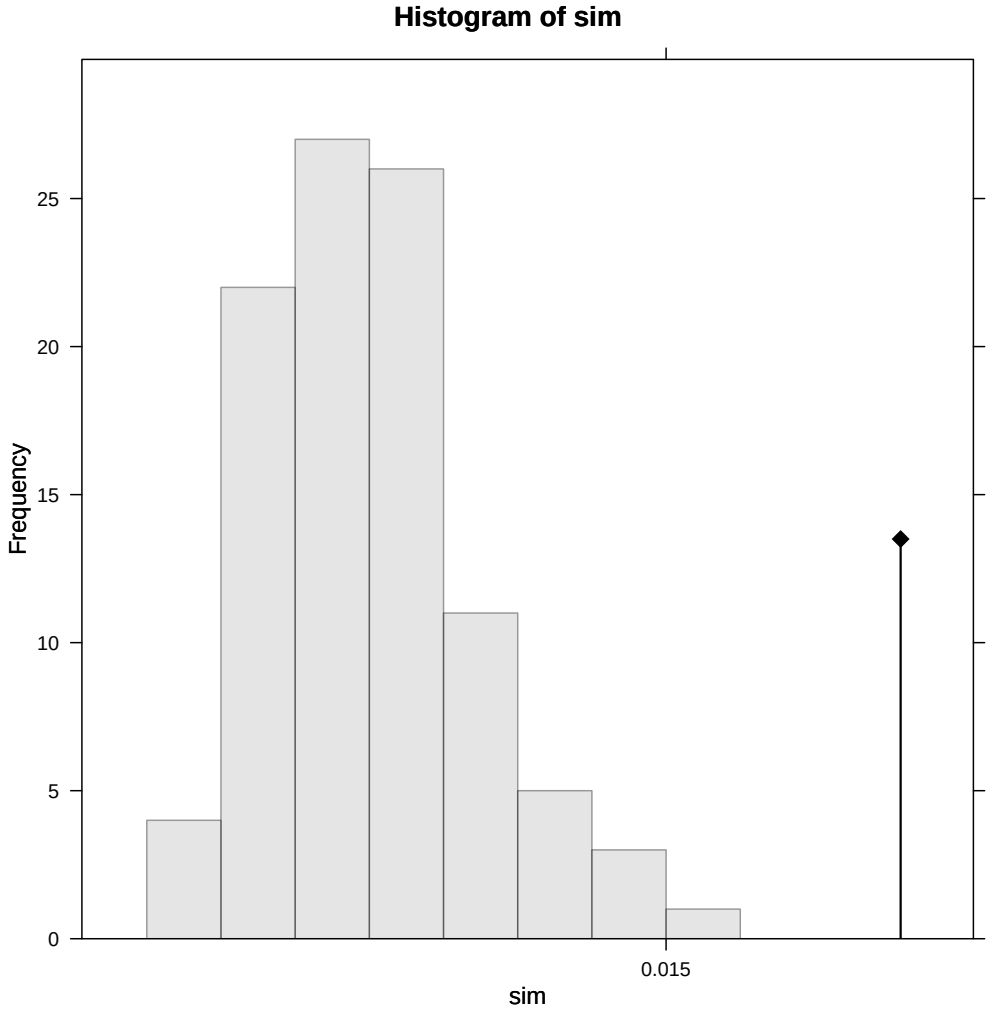


The resulting figure represents eigenvalues of sPCA (denoted λ_i with $i = 1, \ldots, r$, where λ_1 is the highest positive eigenvalue, and λ_r is the highest negative eigenvalue) according the their variance and Moran's I components. These eigenvalues are contained inside a rectangle indicated in dashed lines. The maximum attainable variance by a linear combination of alleles is the one from an ordinary PCA, indicated by the vertical dashed line on the right. The two horizontal dashed lines indicate the range of variation of Moran's I, given the spatial weighting matrix that was used. This figure is useful to assess whether a given score of entities contains relatively enough variability and spatial structuring to be interpreted. For instance, here, λ_1 clearly is the largest eigenvalue in terms of variance and of spatial autocorrelation, and can be well distinguished from all the other eigenvalues. Hence, only the first global structure, associated to λ_1, should be interpreted.

The global and local tests proposed in [1] can be used to reinforce the decision of interpreting or not interpreting global and local structures. Each test can detect the presence of one kind of structure. We can apply them to the object `obj`, used in our sPCA:

```
myGtest <- global.rtest(obj$tab,mySpca$lw,nperm=99)
myGtest
```

```
## Monte-Carlo test
## Call: global.rtest(X = obj$tab, listw = mySpca$lw, nperm = 99)
##
## Observation: 0.01658103
##
## Based on 99 replicates
## Simulated p-value: 0.01
## Alternative hypothesis: greater
##
##      Std.Obs  Expectation     Variance
## 4.987757e+00 1.300095e-02 5.151976e-07

plot(myGtest)
```

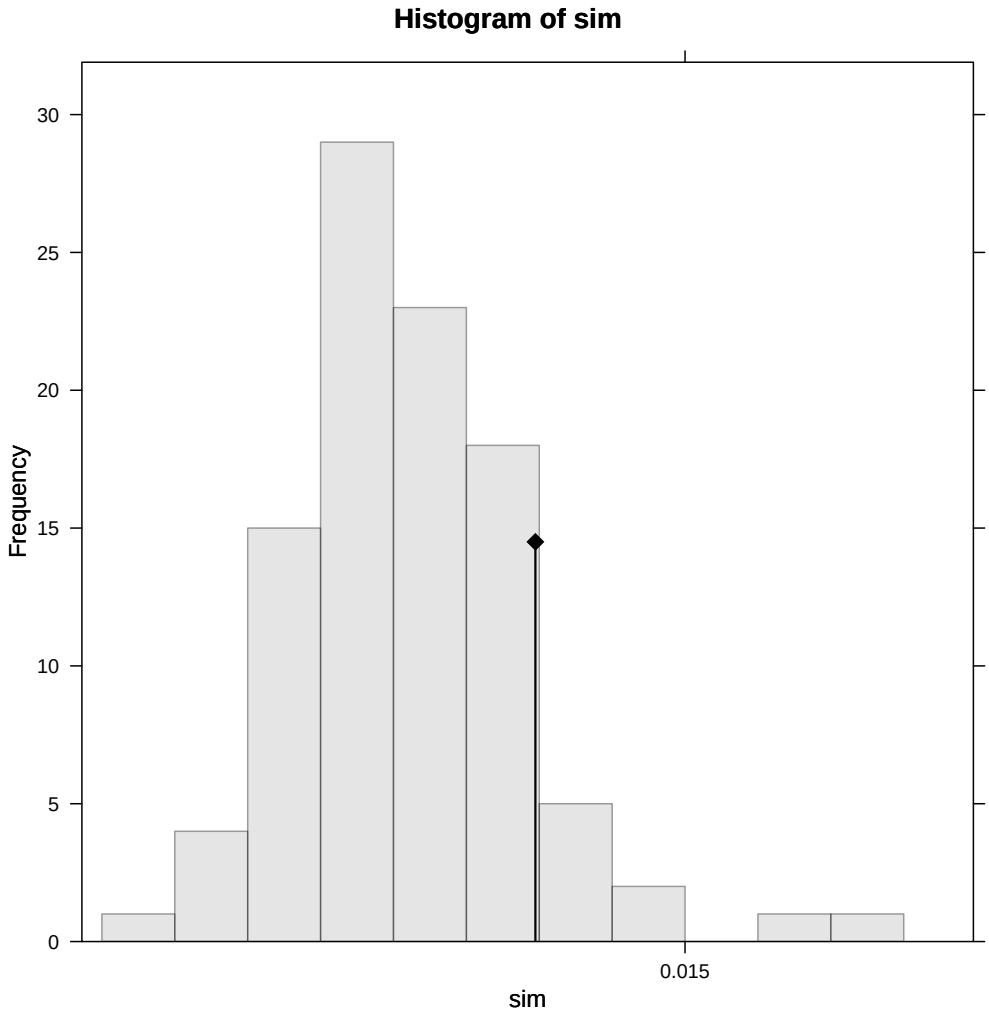


The produced object is a `randtest` object (see `?randtest`), which is the class of objects for Monte-Carlo tests in the *ade4* package. As shown, such object can be plotted using a `plot` function: the resulting figure shows an histogram of permuted test statistics and indicates the observed statistics by a black dot and a segment. Here, the plot clearly shows that the oberved test statistic is larger than most simulated values, leading to a likely rejection of

the null hypothesis of absence of spatial structure. Note that because 99 permutations were used, the p-value cannot be lower than 0.01. In practice, more permutations should be used (like 999 or 9999 for results intended to be published).

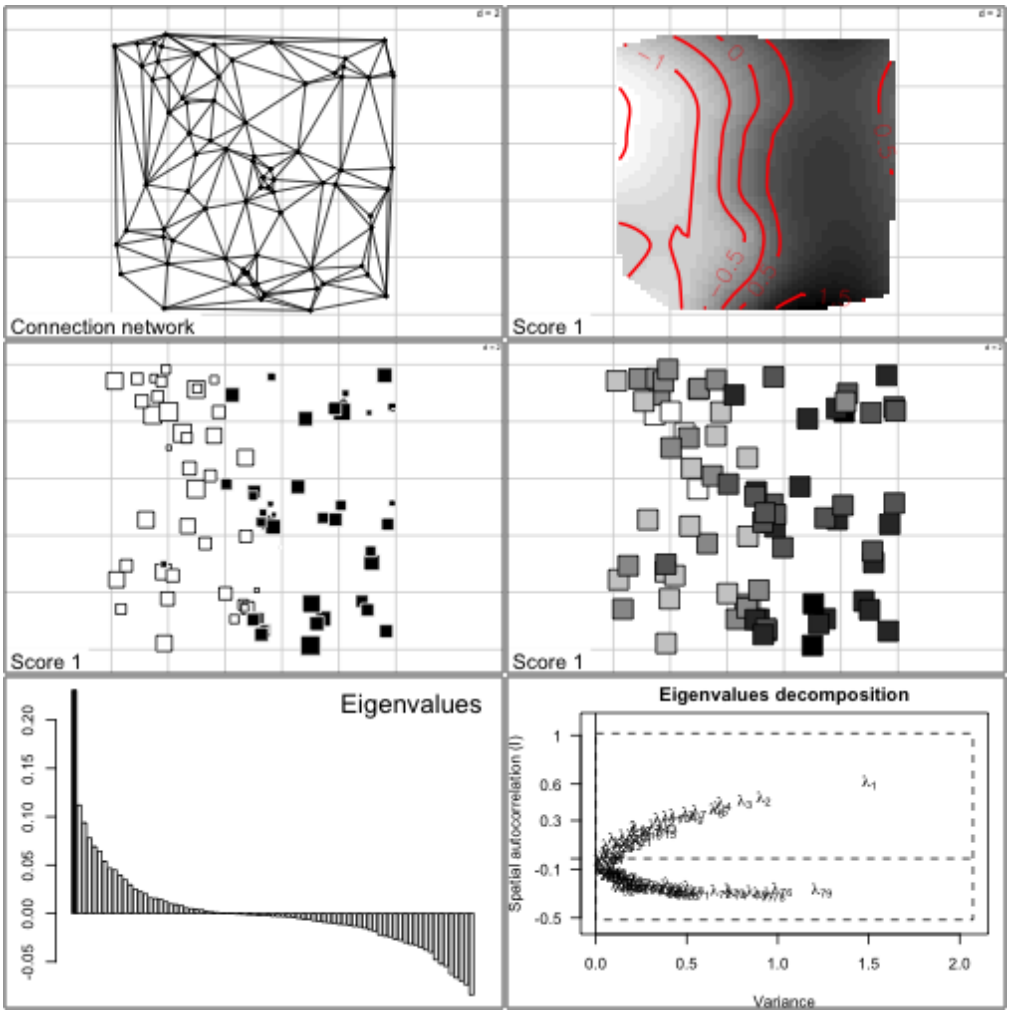
The same can be done with the local test, which here we do not expect to be significant:

```
myLtest <- local.rtest(obj$tab,mySpca$lw,nperm=99)
myLtest

## Monte-Carlo test
## Call: local.rtest(X = obj$tab, listw = mySpca$lw, nperm = 99)
##
## Observation: 0.01397349
##
## Based on 99 replicates
## Simulated p-value: 0.1
## Alternative hypothesis: greater
##
##      Std.Obs  Expectation     Variance
## 1.126504e+00 1.309143e-02 6.131067e-07

plot(myLtest)
```



Once we have an idea of which structures shall be interpreted, we can try to visualize spatial genetic patterns. There are several ways to do so. The first, most simple approach is through the function plot (see `?plot.spca`):

```
plot(mySpca)
```

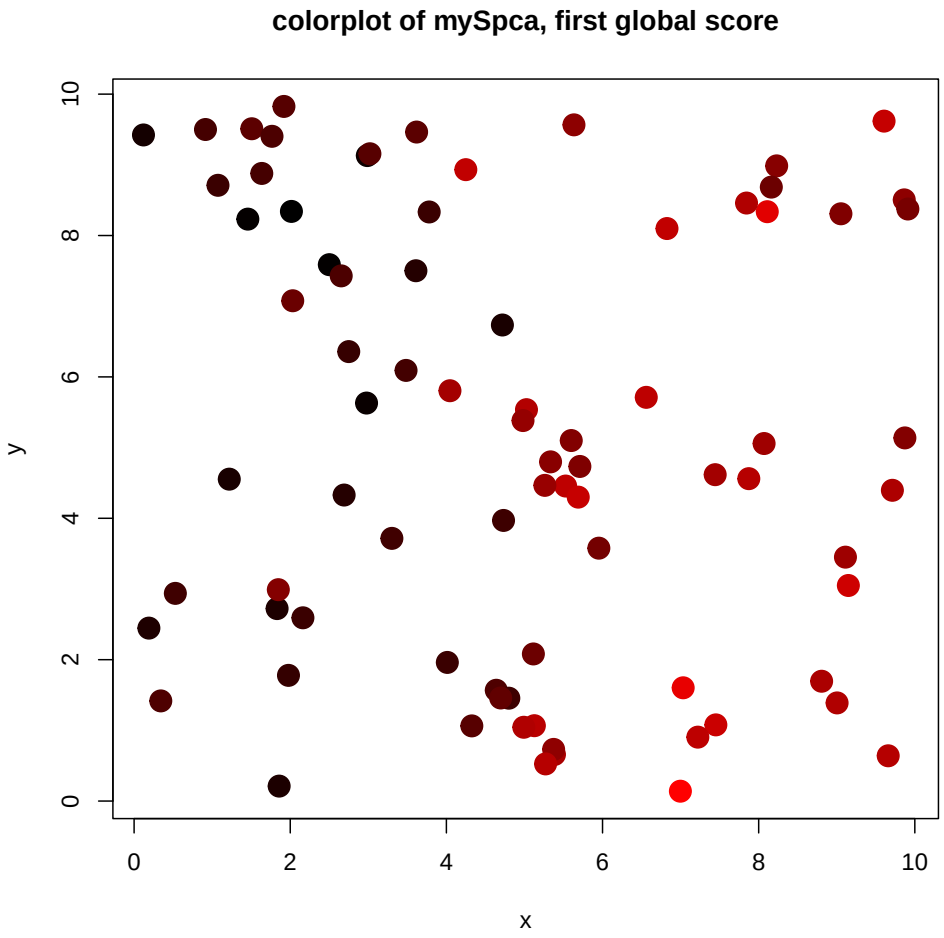


This figure displays various information, that we detail from the top to bottom and from left to right (also see `?plot.spca`). The first plot shows the connection network that was used to define spatial weightings. The second, third, and fourth plots are different representations of a score of entities in space, the first global score being the default (argument `axis`). In each, the values of scores (`$li[,axis]` component of the `spca` object) are represented using black and white symbols (a variant being grey levels): white for negative values, and black for positive values. The second plot is a local interpolation of scores (function `s.image` in *ade4*), using grey levels, with contour lines. The closer the contour lines are from each other, the stepest the genetic differentiation is. The third plot uses different sizes of squares to represent different absolute values (`s.value` in *ade4*): large black squares are well differentiated from large white squares, but small squares are less differentiated. The fourth plot is a variant using grey levels (`s.value` in *ade4*, with 'greylevel' method). Here, all the three representations of the first global score show that genotypes are splitted in two genetical clusters, one in the west (or left) and one in the east (right). The last two plots of the `plot.spca` function are the two already seen displays of eigenvalues.

While the default `plot` function for `spca` objects provides a useful summary of the results, more flexible tools are needed e.g. to map the principal components onto the geographic

space. This can be achieved using the `colorplot` function. This function can summarize up to three scores at the same time by translating each score into a channel of color (red, green, and blue). The obtained values are used to compose a color using the RGB system. See `?colorplot` for details about this function. The original idea of such representation is due to [8]. Despite the `colorplot` clearly is more powerful to represent more than one score on a single map, we can use it to represent the first global structure that was retained in `mySpca`:

```
colorplot(mySpca,cex=3,main="colorplot of mySpca, first global score")
```

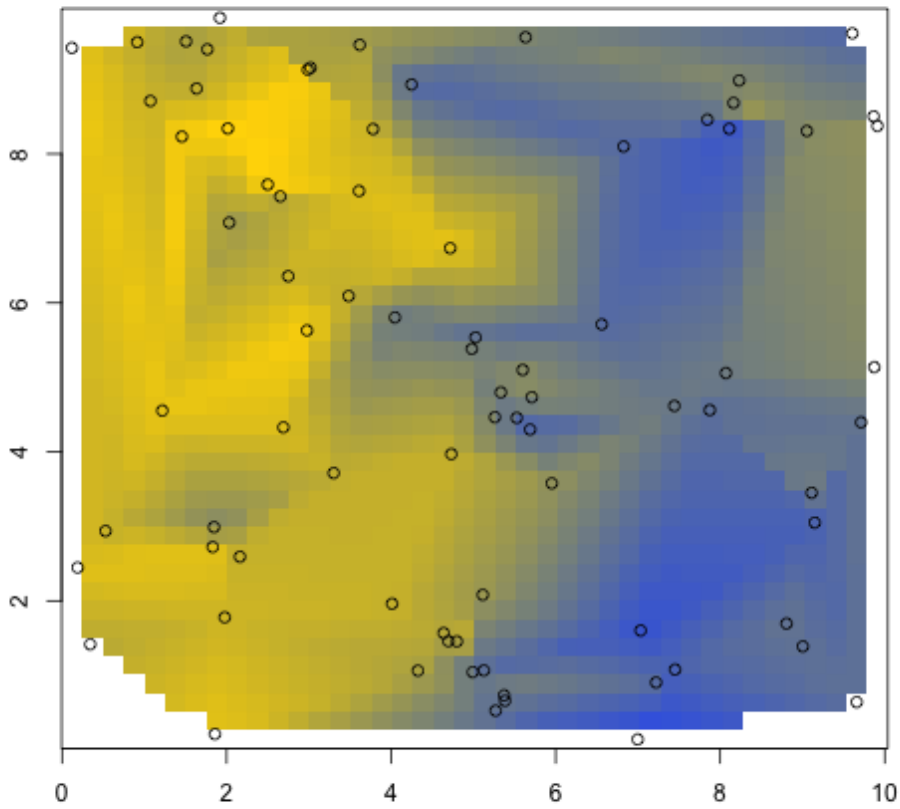


See examples in `?colorplot` and `?spca` for more examples of applications of colorplot to represent sPCA scores.

Another common practice is interpolating principal components to get maps of genetic clines. Note that it is crucial to perform this interpolation after the analysis, and not before, which would add artefactual structures to the data. Interpolation is easy to realize using `interp` from the `akima` package, and `image`, or `filled.contour` to display the results:

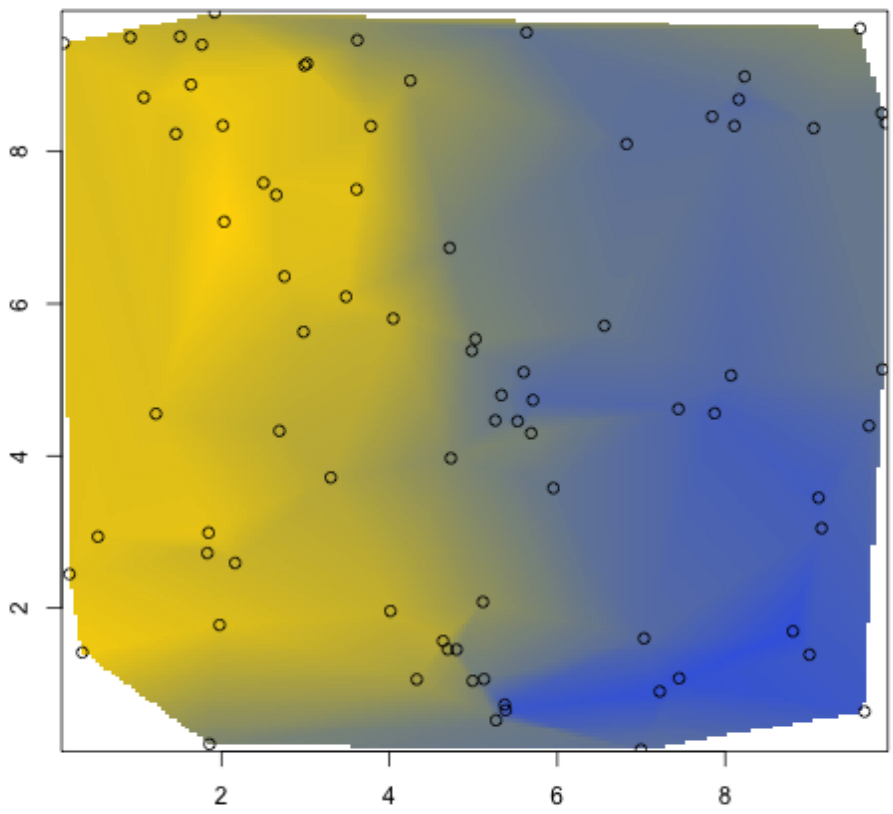
```
library(akima)
x <- other(obj)$xy[,1]
y <- other(obj)$xy[,2]
```

```
temp <- interp(x, y, mySpca$li[,1])
image(temp, col=azur(100))
points(x,y)
```



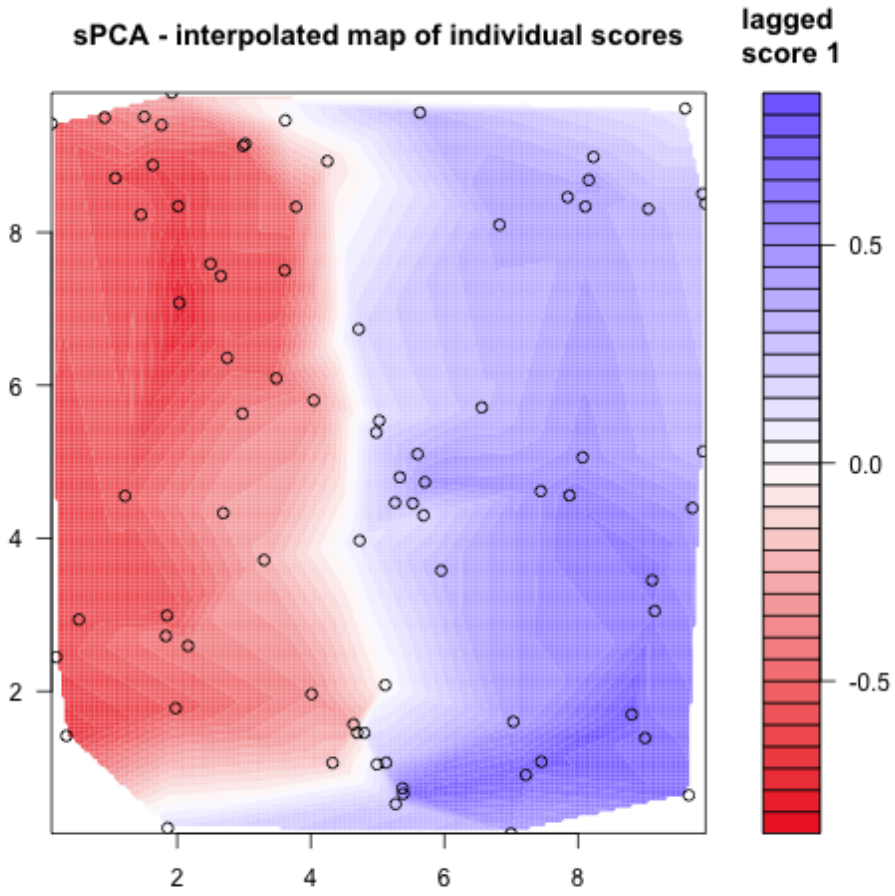
Note that for better clarity, we can use the lagged principal scores (`$ls`) rather than the original scores (`$li`); we also achieve a better resolution using specific interpolated coordinates:

```
interpX <- seq(min(x),max(x),le=200)
interpY <- seq(min(y),max(y),le=200)
temp <- interp(x, y, mySpca$ls[,1], xo=interpX, yo=interpY)
image(temp, col=azur(100))
points(x,y)
```



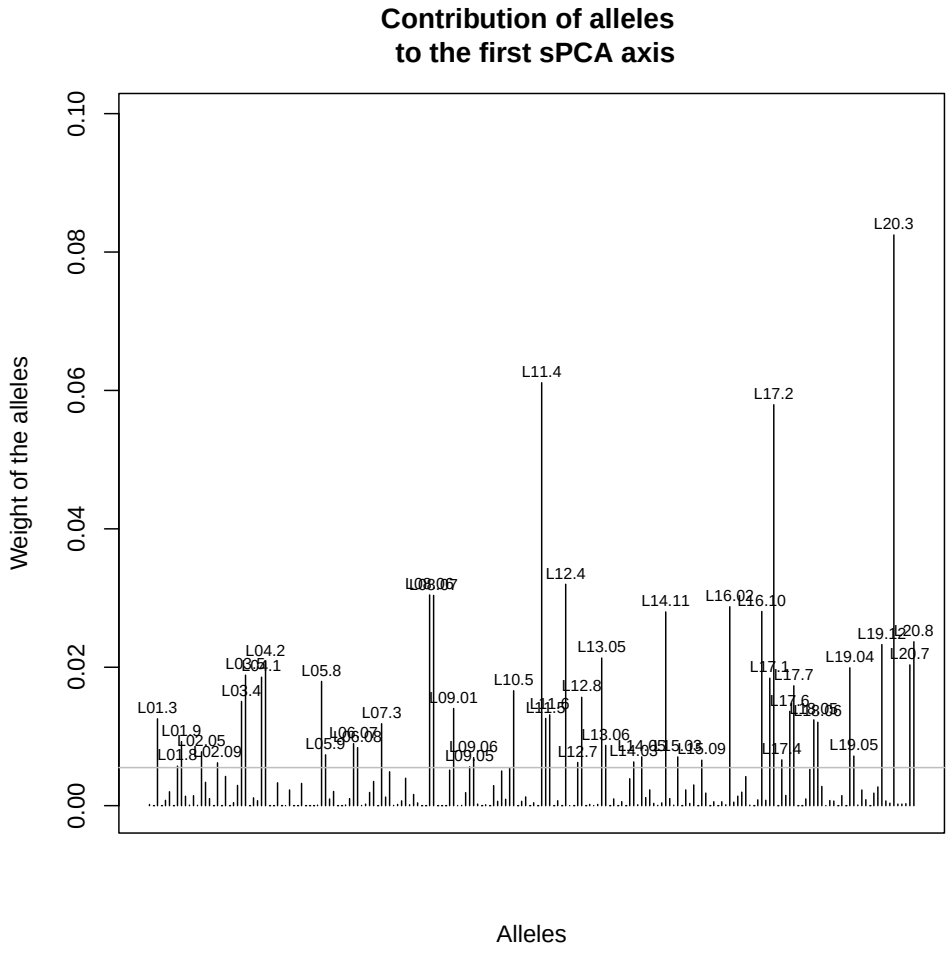
Alternatively, `filled.contour` can be used for the display, and a customized color palette can be specified:

```
myPal <- colorRampPalette(c("firebrick2", "white", "lightslateblue"))
annot <- function(){
    title("sPCA - interpolated map of individual scores")
    points(x,y)
}
filled.contour(temp, color.pal=myPal, nlev=50,
               key.title=title("lagged \nscore 1"), plot.title=annot())
```



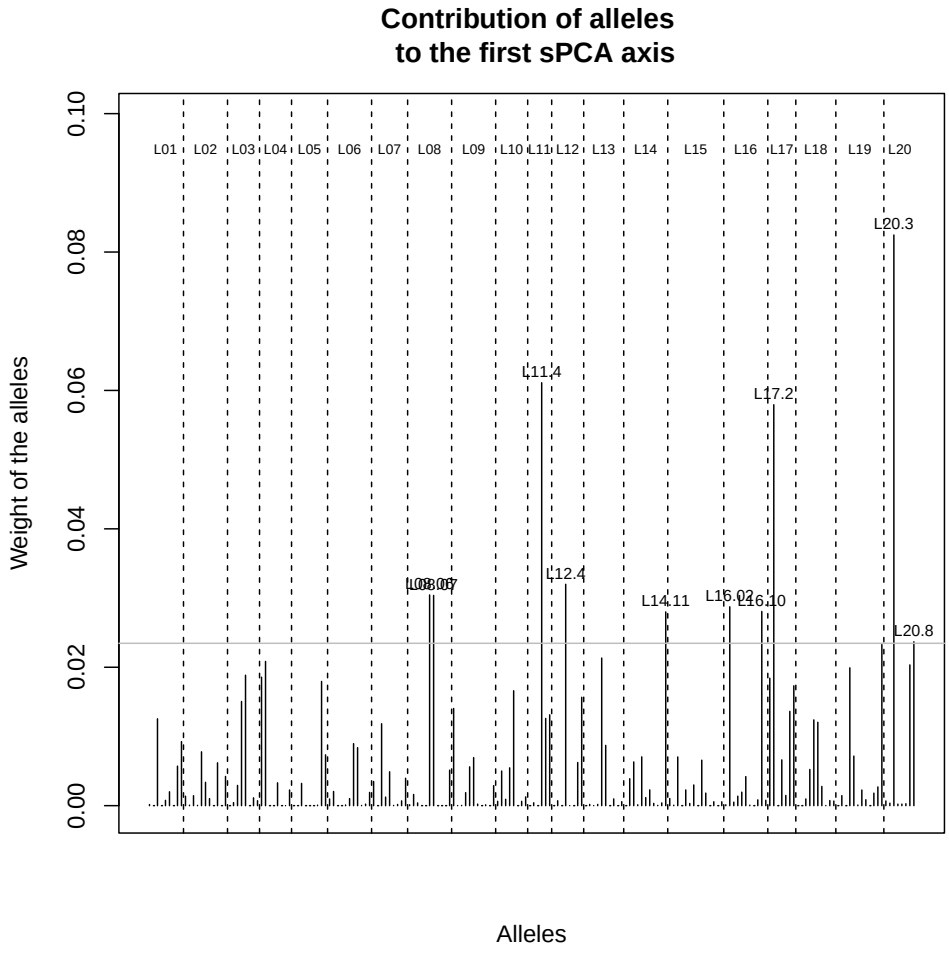
Besides assessing spatial patterns, it is sometimes valuable to assess which alleles actually exhibit the structure of interest. In sPCA, the contribution of alleles to a specific structure is given by the corresponding squared loading. We can look for the alleles contributing most to e.g. the first axis of sPCA, using the function `loadingplot` (see `?loadingplot` for a description of the arguments):

```
myLoadings <- mySpca$c1[,1]^2
names(myLoadings) <- rownames(mySpca$c1)
loadingplot(myLoadings, xlab="Alleles",
            ylab="Weight of the alleles",
            main="Contribution of alleles \n to the first sPCA axis")
```



See `?loadingplot` for more information about this function, in particular for the definition of the threshold value above which alleles are annotated. Note that it is possible to also separate the alleles by markers, using the `fac` argument, to assess if all markers have comparable contributions to a given structure. In our case, we would only have to specify `fac=obj@loc.fac`; also note that `loadingplot` invisibly returns information about the alleles whose contribution is above the threshold. For instance, to identify the 5% of alleles with the greatest contributions to the first global structure in `mySpca`, we need:

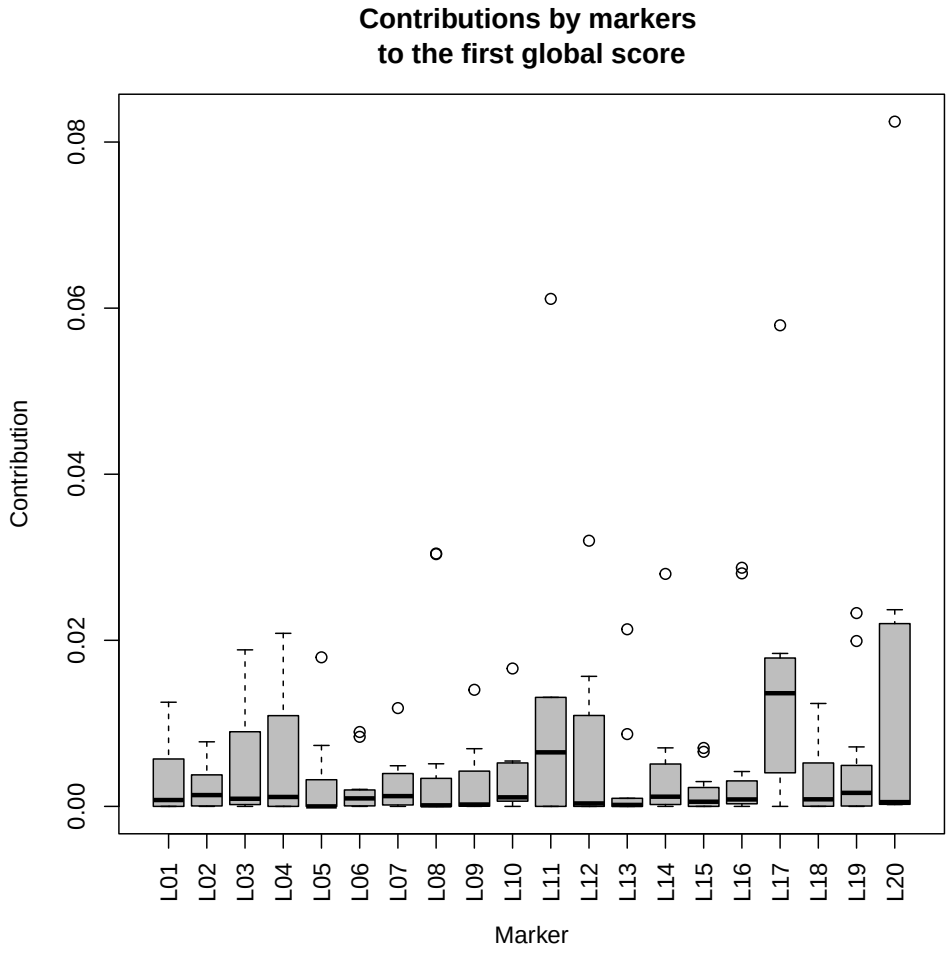
```
temp <- loadingplot(myLoadings, threshold=quantile(myLoadings, 0.95),
                    xlab="Alleles",ylab="Weight of the alleles",
                    main="Contribution of alleles \n to the first sPCA axis",
                    fac=obj$loc.fac, cex.fac=0.6)
```




```
temp

## $threshold
##        95%
## 0.02345973
##
## $var.names
##  [1] "L08.06" "L08.07" "L11.4"  "L12.4"  "L14.11" "L16.02" "L16.10" "L17.2"  "L20.3"
##
## $var.idx
## L08.06 L08.07  L11.4  L12.4 L14.11 L16.02 L16.10  L17.2  L20.3  L20.8
##     71     72     99    105    130    146    154    157    187    192
##
## $var.values
##     L08.06     L08.07      L11.4      L12.4     L14.11     L16.02     L16.10      L17
## 0.03044687 0.03037709 0.06111338 0.03199067 0.02799529 0.02873923 0.02806079 0.057932
```

But to assess the average contribution of each marker, the boxplot probably is a better tool:

```
boxplot(myLoadings~obj$loc.fac, las=3, ylab="Contribution", xlab="Marker",
        main="Contributions by markers \nto the first global score", col="grey")
```



2 Case study: spatial genetic structure of the chamois in the Bauges mountains

The chamois (*Rupicapra rupicapra*) is a conserved species in France. The Bauges mountains is a protected area in which the species has been recently studied. One of the most important questions for conservation purposes relates to whether individuals from this area form a single reproductive unit, or whether they are structured into sub-groups, and if so, what causes are likely to induce this structuring.

While field observations are very scarce and do not allow to answer this question, genetic data can be used to tackle the issue, as departure from panmixia should result in genetic structuring. The dataset *rupica* contains 335 georeferenced genotypes of Chamois from the Bauges mountains for 9 microsatellite markers, which we propose to analyse.

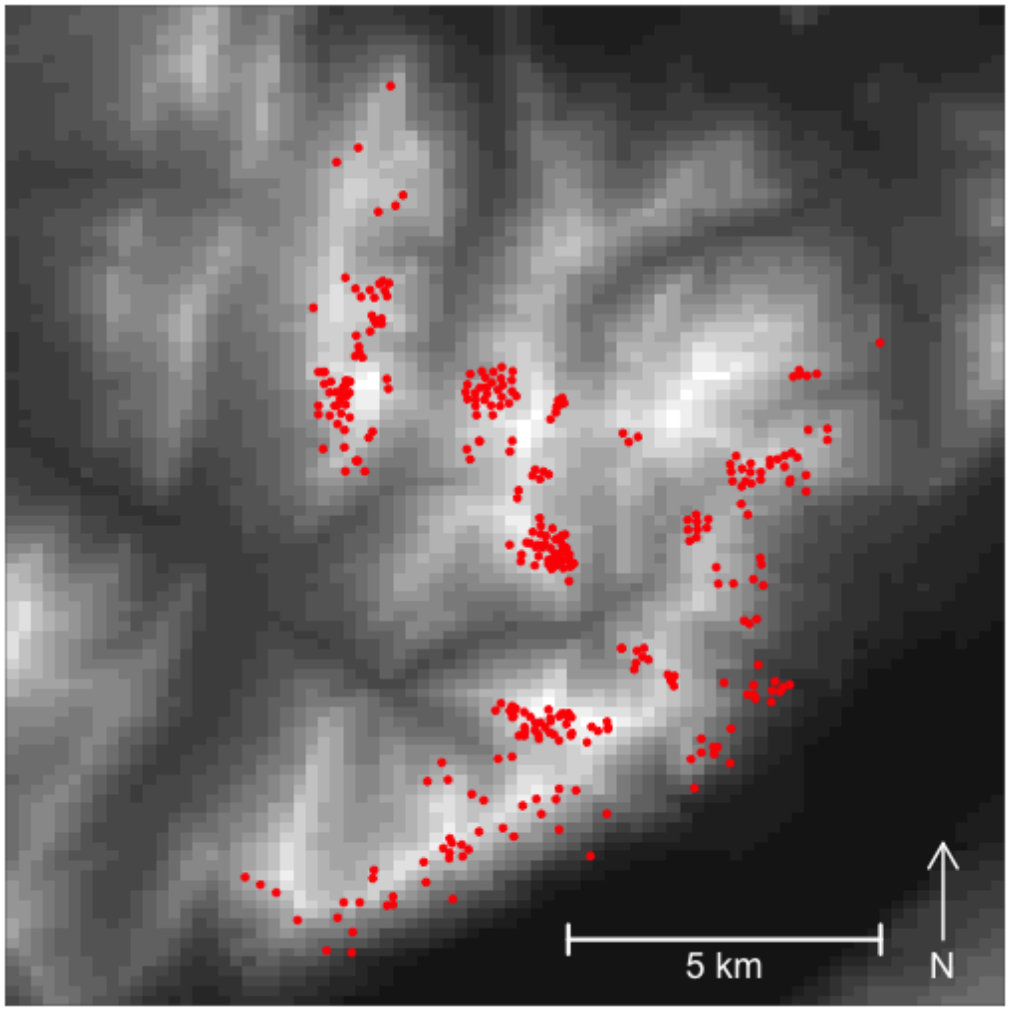
2.1 An overview of the data

We first load the data:

```
data(rupica)
rupica

## /// GENIND OBJECT /////////
##
##  // 335 individuals; 9 loci; 55 alleles; size: 515.9 Kb
##
##  // Basic content
##    @tab:  335 x 55 matrix of allele counts
##    @loc.n.all: number of alleles per locus (range: 4-10)
##    @loc.fac: locus factor for the 55 columns of @tab
##    @all.names: list of allele names for each locus
##    @ploidy: ploidy of each individual  (range: 2-2)
##    @type:  codom
##    @call: NULL
##
##  // Optional content
##    @other: a list containing: xy  mnt  showBauges  image.asc  contour.asc
```

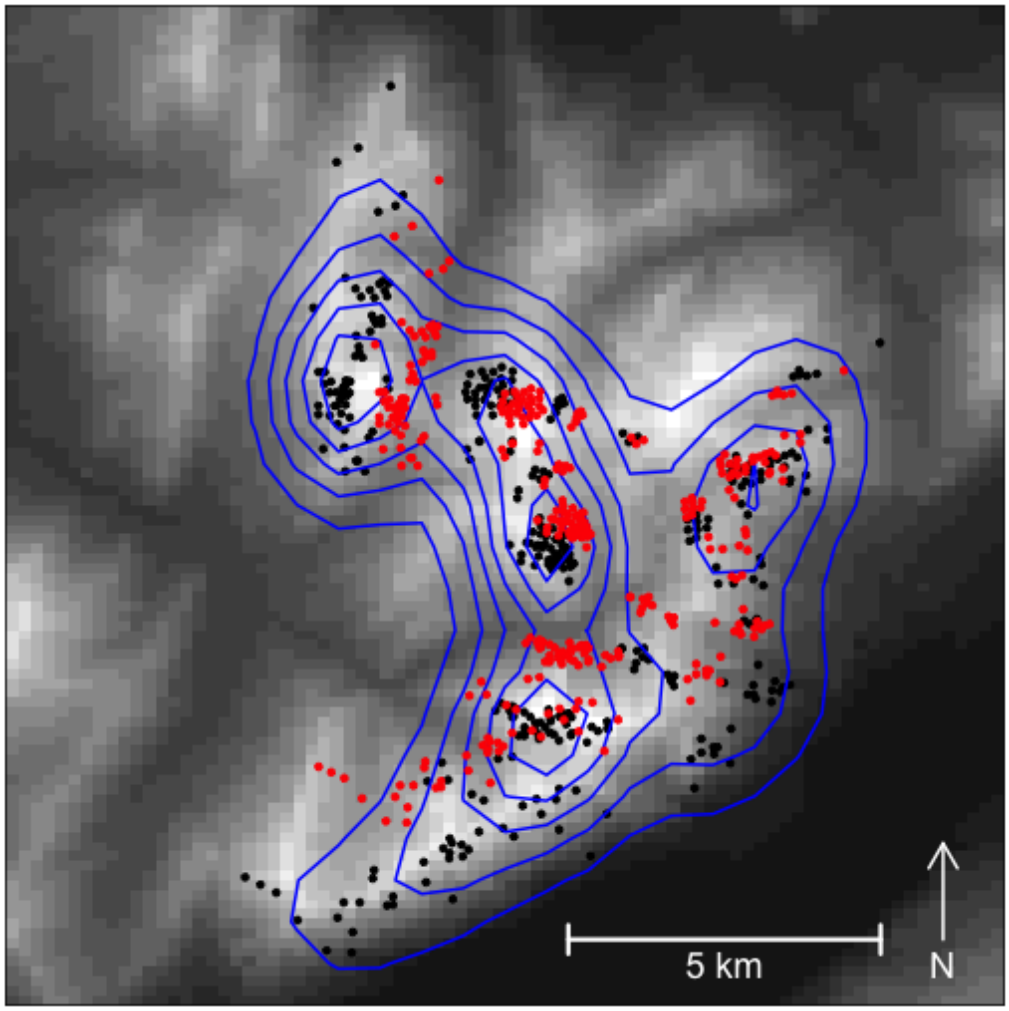
`rupica` is a `genind` object, that is, the class of objects storing genotypes (as opposed to population data) in *adegenet*. `rupica` also contains topographic information about the sampled area, which can be displayed by calling `rupica$other$showBauges`. The spatial distribution of the sampling can be displayed as follows:

```
rupica$other$showBauges()
points(rupica$other$xy, col="red",pch=20)
```



This spatial distribution is clearly not random, but seems arranged into loose clusters. However, superimposed samples can bias our visual assessment of the spatial clustering. Use a two-dimensional kernel density estimation (function `s.kde2d`) to overcome this possible issue.

```
rupica$other$showBauges()
s.kde2d(rupica$other$xy,add.plot=TRUE)
points(rupica$other$xy, col="red",pch=20)
```

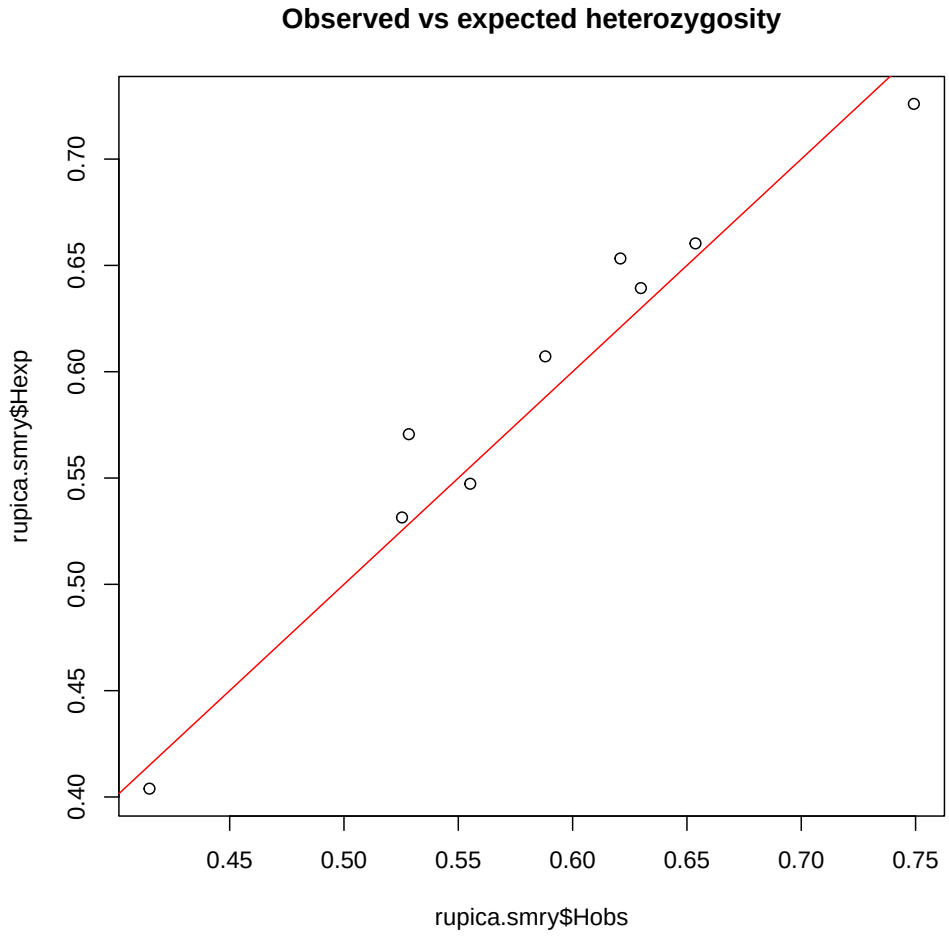


Unfortunately, geographical clustering is not strong enough to assign unambiguously each individual to a group. Therefore, we need to carry all analyses at the individual level, which precludes the use of most population genetics tools.

2.2 Summarising the genetic diversity

As a prior clustering of genotypes is not known, we cannot employ usual F_{ST}-based approaches to detect genetic structuring. However, genetic structure could still result in a deficit of heterozygosity. Use the `summary` of `genind` objects to compare expected and observed heterozygosity:

```
rupica.smry <- summary(rupica)
plot(rupica.smry$Hobs, rupica.smry$Hexp, main="Observed vs expected heterozygosity")
abline(0,1,col="red")
```



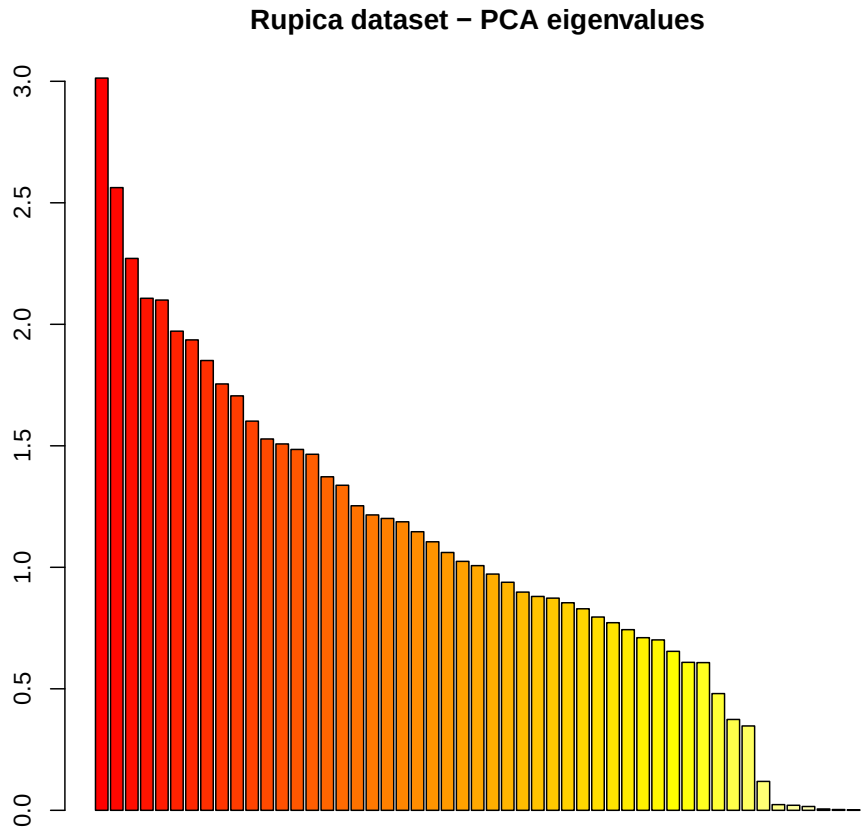
The red line indicate identity between both quantities. Observed heterozygosity do not seem to deviate massively from theoretical expectations. This is confirmed by a classical pairwise t-test::

```
t.test(rupica.smry$Hexp, rupica.smry$Hobs,paired=TRUE,var.equal=TRUE)

##
##  Paired t-test
##
## data:  rupica.smry$Hexp and rupica.smry$Hobs
## t = 1.1761, df = 8, p-value = 0.2734
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.007869215  0.024249885
## sample estimates:
## mean of the differences
##             0.008190335
```

We can seek a global picture of the genetic diversity among genotypes using a Principal Component Analysis (PCA, function `dudi.pca` in the `ade4` package). The analysis is performed on a table of alleles frequencies, obtained by `tab`.

The function `dudi.pca` displays a barplot of eigenvalues and asks for a number of retained principal components:

```
rupica.X <- tab(rupica, freq = TRUE, NA.method = "mean")
rupica.pca1 <- dudi.pca(rupica.X, scale = TRUE, scannf = FALSE, nf = 2)
barplot(rupica.pca1$eig, main = "Rupica dataset - PCA eigenvalues",
        col = heat.colors(length(rupica.pca1$eig)))
```



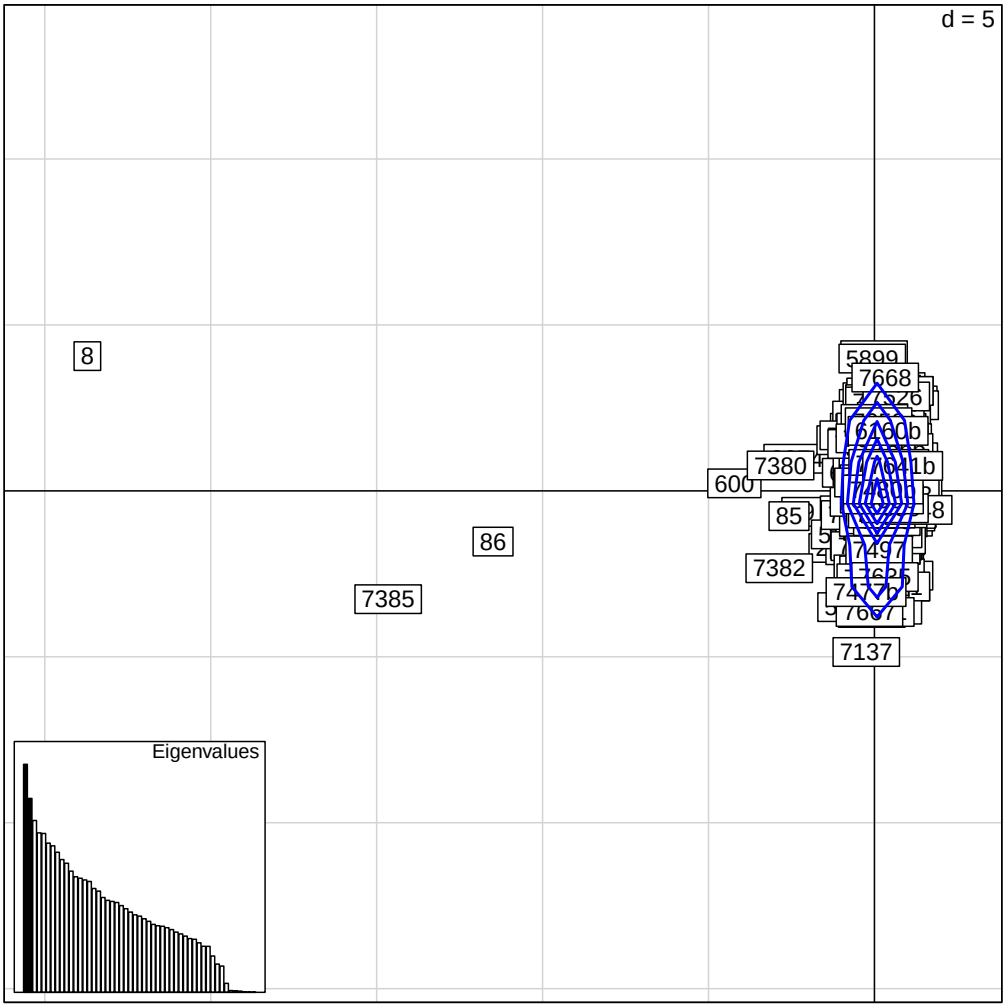
The output produced by `dudi.pca` is a `dudi` object. A `dudi` object contains various information; in the case of PCA, principal axes (loadings), principal components (synthetic variable), and eigenvalues are respectively stored in `$c1`, `$li`, and `$eig` slots. Here is the content of the PCA:

```
rupica.pca1

## Duality diagramm
## class: pca dudi
## $call: dudi.pca(df = rupica.X, scale = TRUE, scannf = FALSE, nf = 2)
##
## $nf: 2 axis-components saved
## $rank: 51
## eigen values: 3.013 2.563 2.271 2.107 2.1 ...
##   vector length mode    content
## 1 $cw    55     numeric column weights
## 2 $lw    335    numeric row weights
## 3 $eig   51     numeric eigen values
##
##   data.frame nrow ncol content
## 1 $tab       335  55   modified array
## 2 $li        335  2    row coordinates
## 3 $l1        335  2    row normed scores
## 4 $co        55   2    column coordinates
## 5 $c1        55   2    column normed scores
## other elements: cent norm
```

In general, eigenvalues represent the amount of genetic diversity — as measured by the multivariate method being used — represented by each principal component (PC). An abrupt decrease in eigenvalues is likely to indicate the boundary between true patterns and non-interpretable structures. In this case, the first two PCs may contain some relevant biological signal.

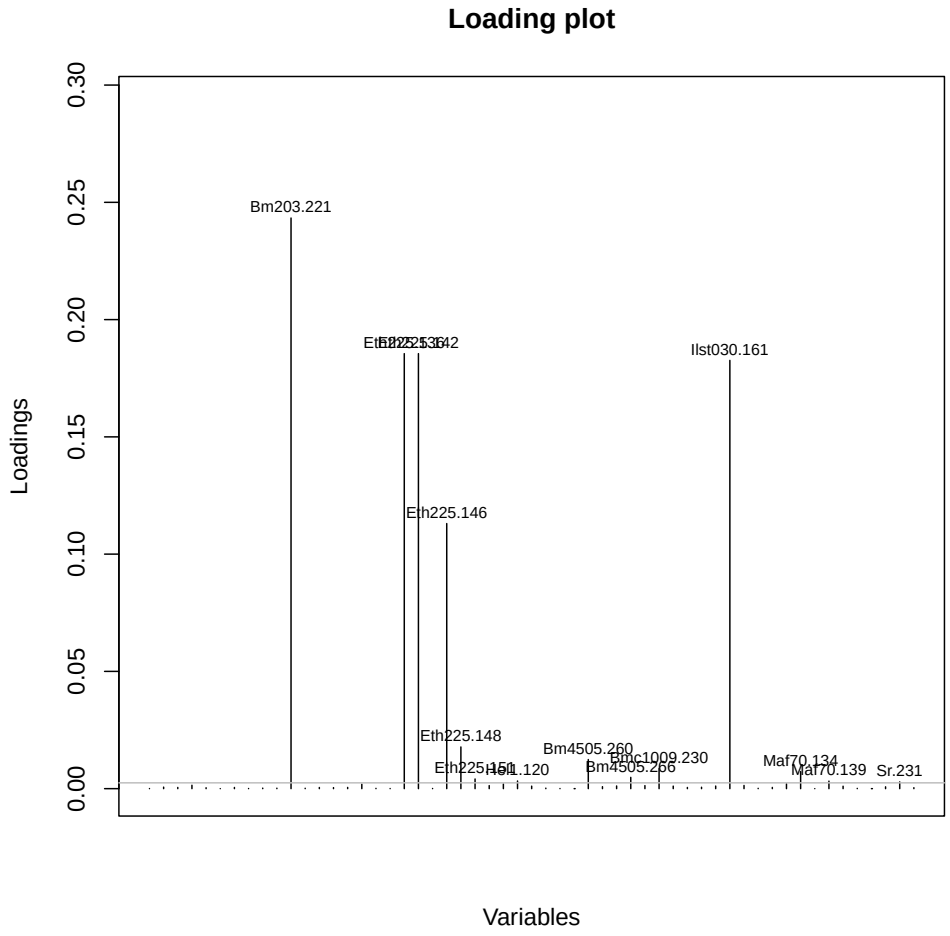
We can use `s.label` to display the two first components of the analysis. Kernel density estimation (`s.kde2d`) is used for a better assessment of the distribution of the genotypes onto the principal axes:

```
s.label(rupica.pca1$li)
s.kde2d(rupica.pca1$li, add.p=TRUE, cpoint=0)
add.scatter.eig(rupica.pca1$eig,2,1,2)
```



This scatterplot shows that the only structure identified by PCA points to a few outliers. `loadingplot` confirms that this corresponds to the possession of a few original alleles:

```
loadingplot(rupica.pca1$c1^2)
```



We can go back to the genotypes for the concerned markers (e.g., Bm203) to check whether the highlighted genotypes are uncommon. `tab` extracts the table of allele frequencies from a `genind` object (restoring original labels for markers, alleles, and individuals):

```
X <- tab(rupica)
class(X)

## [1] "matrix" "array"

dim(X)

## [1] 335  55

bm203.221 <- X[,"Bm203.221"]
table(bm203.221)

## bm203.221
##   0   1
## 331   4
```

Only 4 genotypes possess one copy of the allele 221 of marker bm203 (the second result corresponds to a replaced missing data). Which individuals are they?

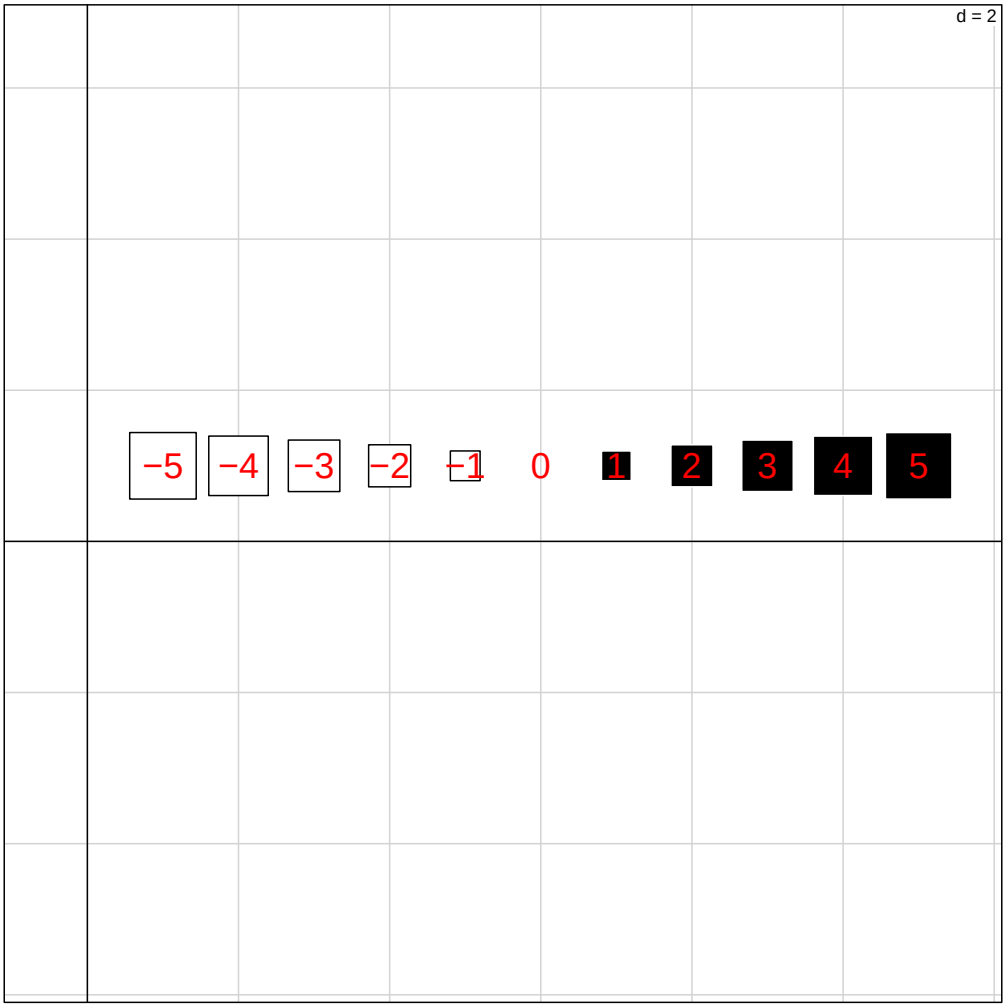
```
rownames(X)[bm203.221 > 0.5]

## [1] "8"    "86"   "600"  "7385"
```

These are indeed our outliers. From the point of view of PCA, this would be the only structure in the data. However, further analyses show that more is to be seen...

2.3 Mapping and testing PCA results

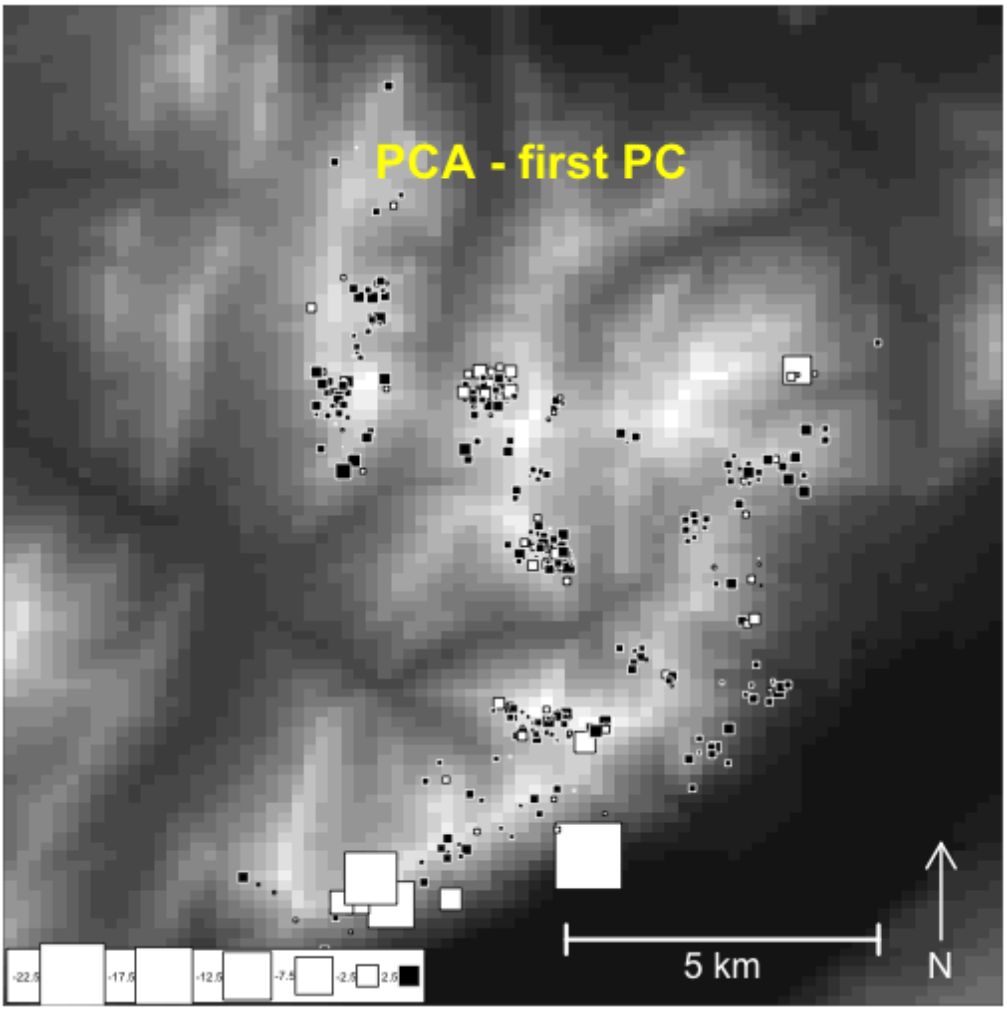
A frequent practice in spatial genetics is mapping the first principal components (PCs) onto the geographic space. *ade4*'s function `s.value` is well-suited to do so, using black and white squares of variable size for positive and negative values. To give a legend for this type of representation:

```
s.value(cbind(1:11,rep(1,11)), -5:5, cleg=0)
text(1:11,rep(1,11), -5:5, col="red",cex=1.5)
```

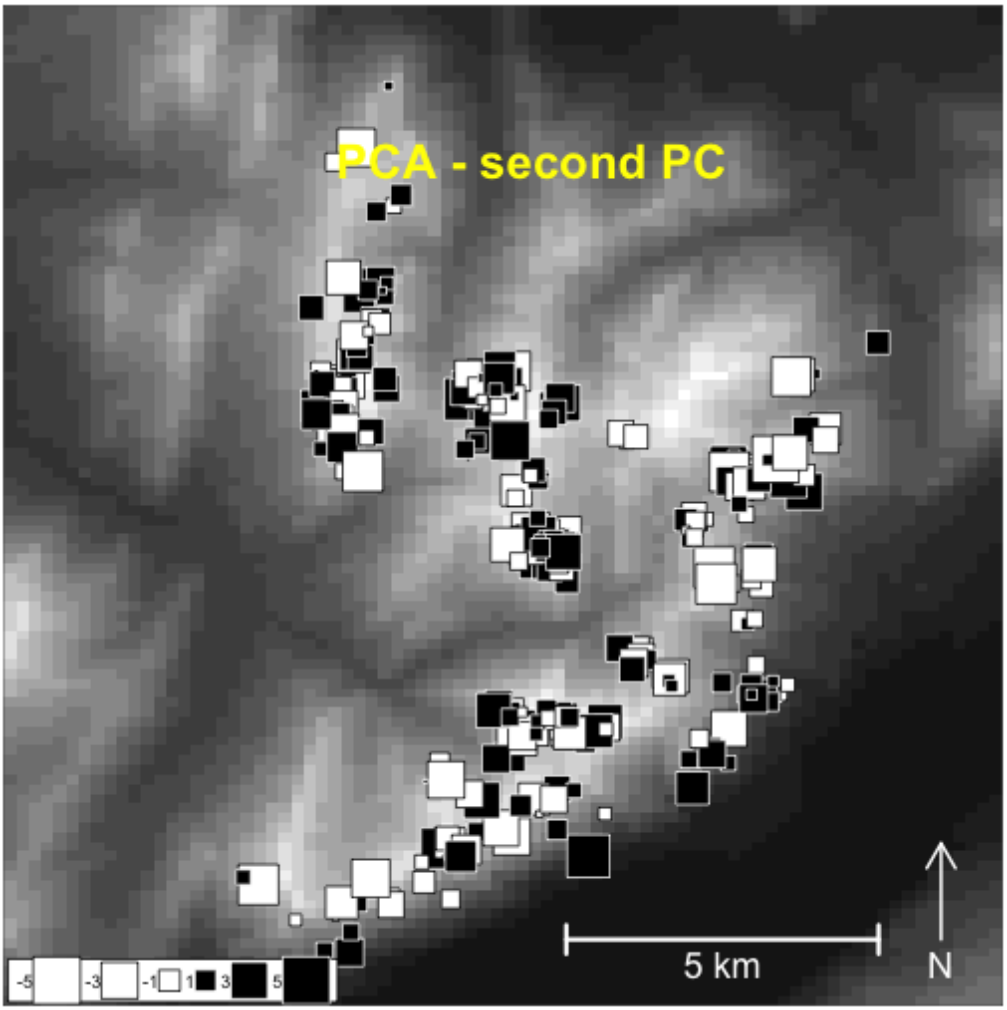


We apply this graphical representation to the first two PCs of the PCA:

```
showBauges <- rupica$other$showBauges
showBauges()
s.value(rupica$other$xy, rupica.pca1$li[,1], add.p=TRUE, cleg=0.5)
title("PCA - first PC",col.main="yellow" ,line=-2, cex.main=2)
```




```
showBauges()
s.value(rupica$other$xy, rupica.pca1$li[,2], add.p=TRUE, csize=0.7)
title("PCA - second PC",col.main="yellow" ,line=-2, cex.main=2)
```



As we can see, none of these PCs seems to display a particular spatial pattern. This visual assessment can be complemented by a test of spatial autocorrelation in these PCs. This can be achieved using Moran's I test. We use *spdep*'s function `moran.mc` to perform these two tests. We first need to define the spatial connectivity between the sampled individuals. For these data, spatial connectivity is best defined as the overlap between home ranges of individuals, modelled as disks with a radius of 1150m. `chooseCN` is used to create the corresponding connection network:

```
rupica.graph <- chooseCN(rupica$other$xy,type=5,d1=0,d2=2300, plot=FALSE,
                         res="listw")
```

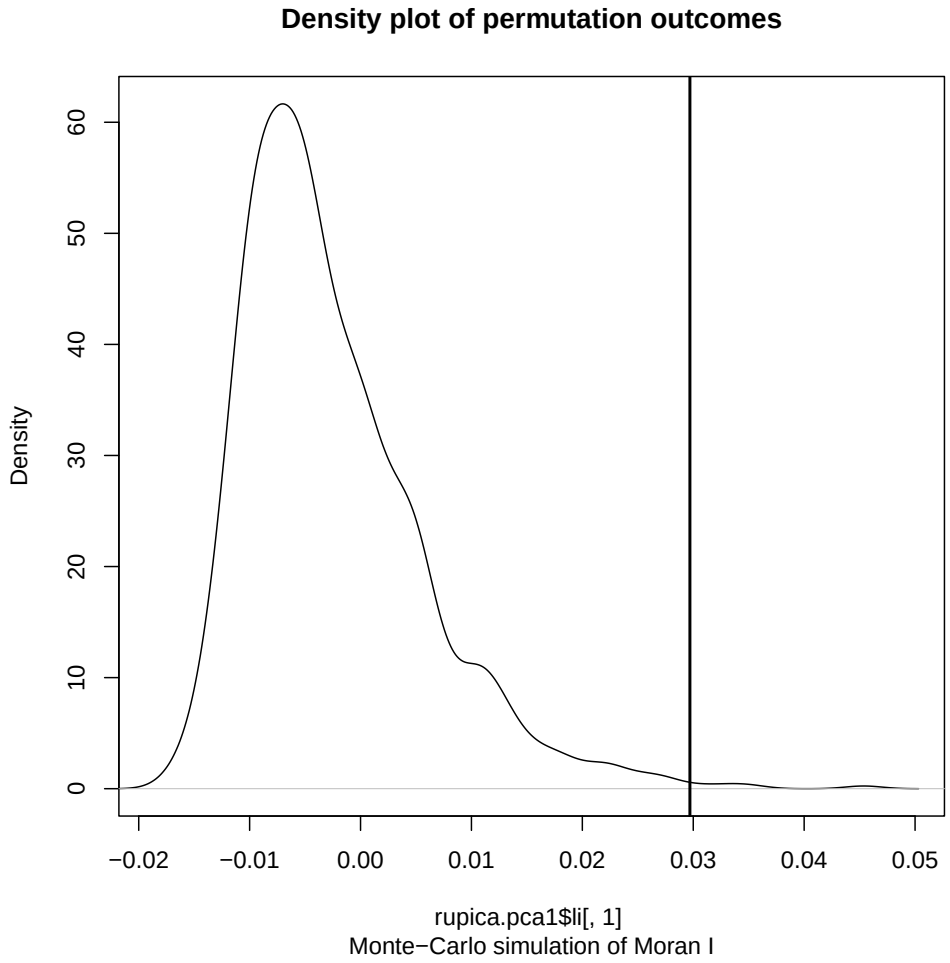
The connection network should ressemble this:

```
rupica.graph

## Characteristics of weights list object:
## Neighbour list object:
## Number of regions: 335
## Number of nonzero links: 18018
```

```
## Percentage nonzero weights: 16.05525
## Average number of links: 53.78507
##
## Weights style: W
## Weights constants summary:
##     n     nn  S0       S1      S2
## W 335 112225 335 15.04311 1352.07

plot(rupica.graph, rupica$other$xy)
title("rupica.graph")
```

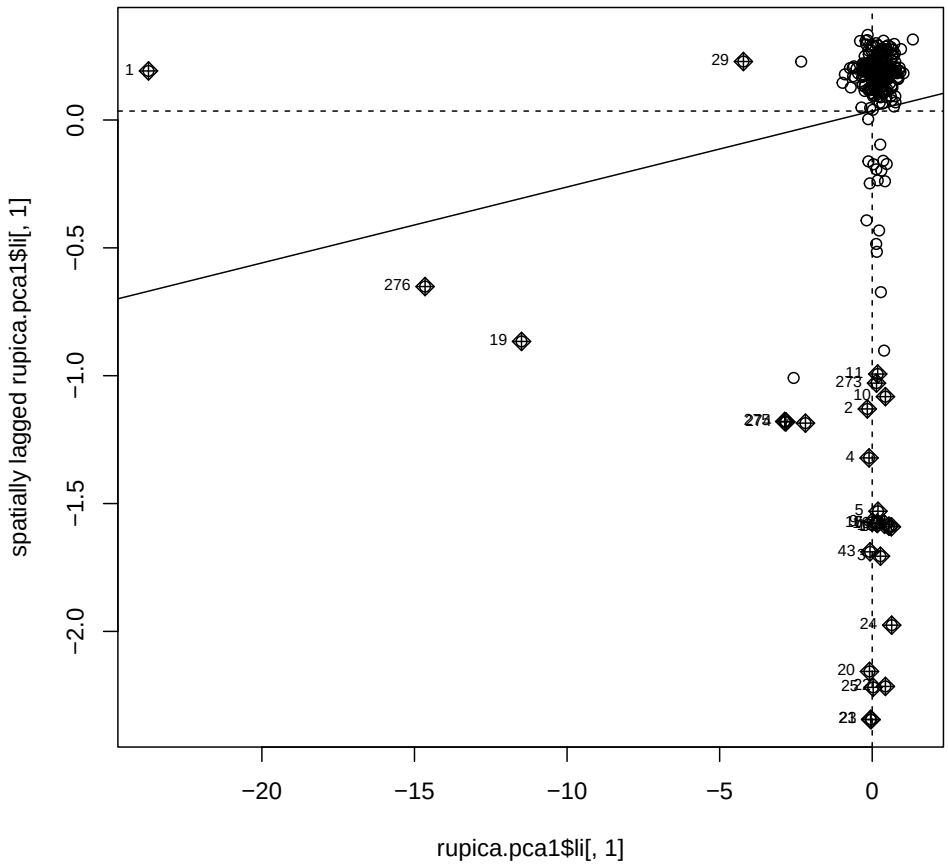
We perform Moran's test for the first two PCs, and plot the results.

```
pc1.mctest <- moran.mc(rupica.pca1$li[,1], rupica.graph, 999)
plot(pc1.mctest)
```



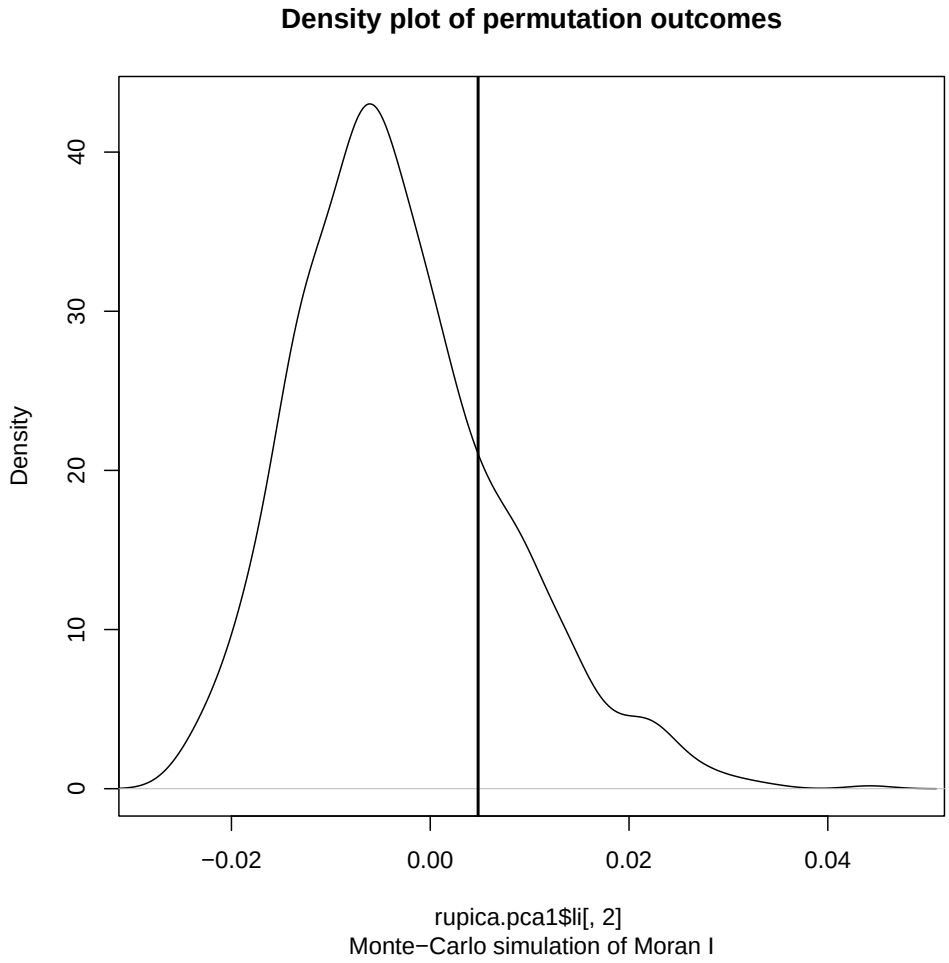
This result is surprisingly significant. Why is this? Moran's plot (`moran.plot`) represents the tested variable against its lagged vector; we apply it to the first PC:

```
moran.plot(rupica.pca1$li[,1], rupica.graph)
```



Positive autocorrelation corresponds to a positive correlation between a variable and its lag vector. Here, we can see that this relation is entirely driven by the previously identified outliers, which turn out to be neighbours. This is therefore a fairly trivial and uninteresting pattern. Results obtained on the second PC are less surprisingly non-significant:

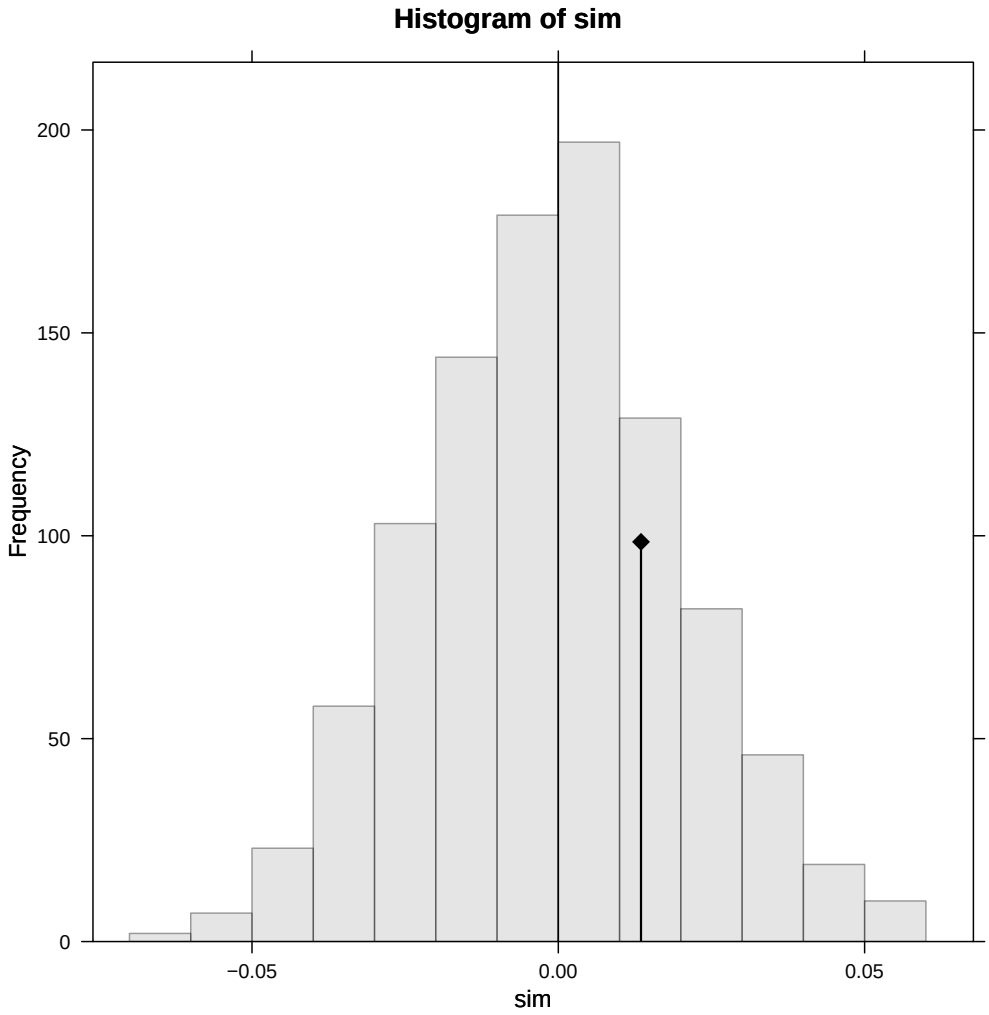
```
pc2.mctest <- moran.mc(rupica.pca1$li[,2], rupica.graph, 999)
plot(pc2.mctest)
```



2.4 Multivariate tests of spatial structure

So far, we have only tested the existence of spatial structures in the first two principal components of a PCA of the data. Therefore, these tests only describe one fragment of the data, and do not encompass the whole diversity in the data. As a complement, we can use Mantel test (`mantel.randtest`) to test spatial structures in the whole data, by assessing the correlation between genetic distances and geographic distances. Pairwise Euclidean distances are computed using `dist`. Perform Mantel test, using the scaled genetic data you used before in PCA, and the geographic coordinates.

```
mtest <- mantel.randtest(dist(rupica.X), dist(rupica$other$xy))
plot(mtest, nclass=30)
```

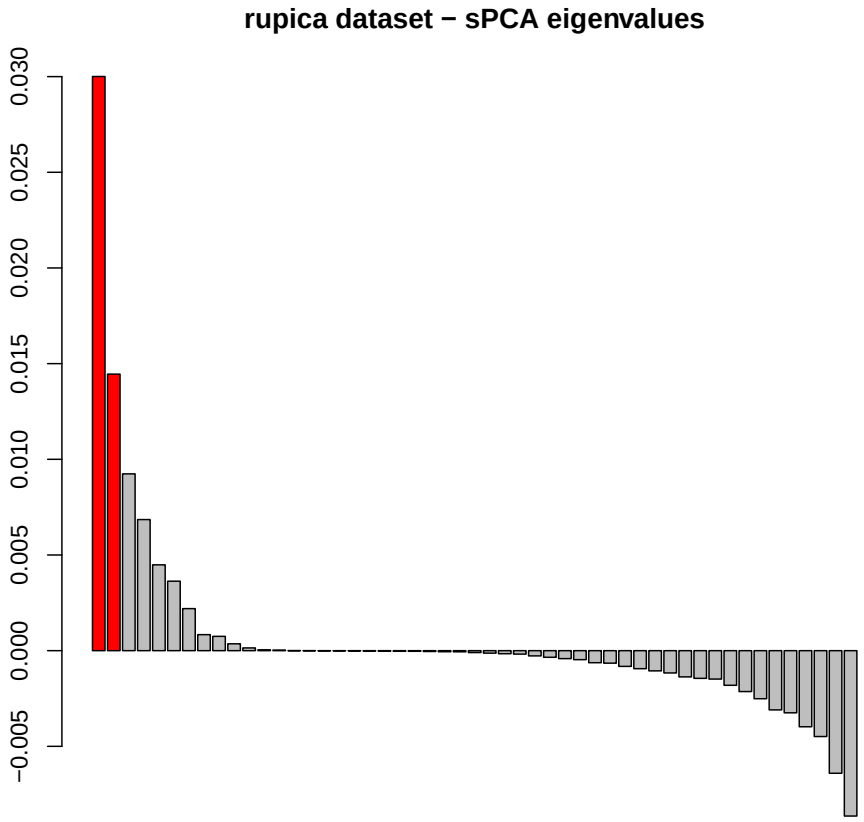


Interestingly, this test turns out to be marginally significant, and would encourage us to look for spatial patterns. This is the role of the spatial Principal Component Analysis.

2.5 Spatial Principal Component Analysis

We apply an sPCA to the `rupica` dataset, using the connection network used previously in Moran's I tests:

```
rupica.spca1 <- spca(rupica, cn=rupica.graph,scannf=FALSE,
                     nfposi=2,nfnega=0)
barplot(rupica.spca1$eig, col=rep(c("red","grey"), c(2,1000)),
        main="rupica dataset - sPCA eigenvalues")
```



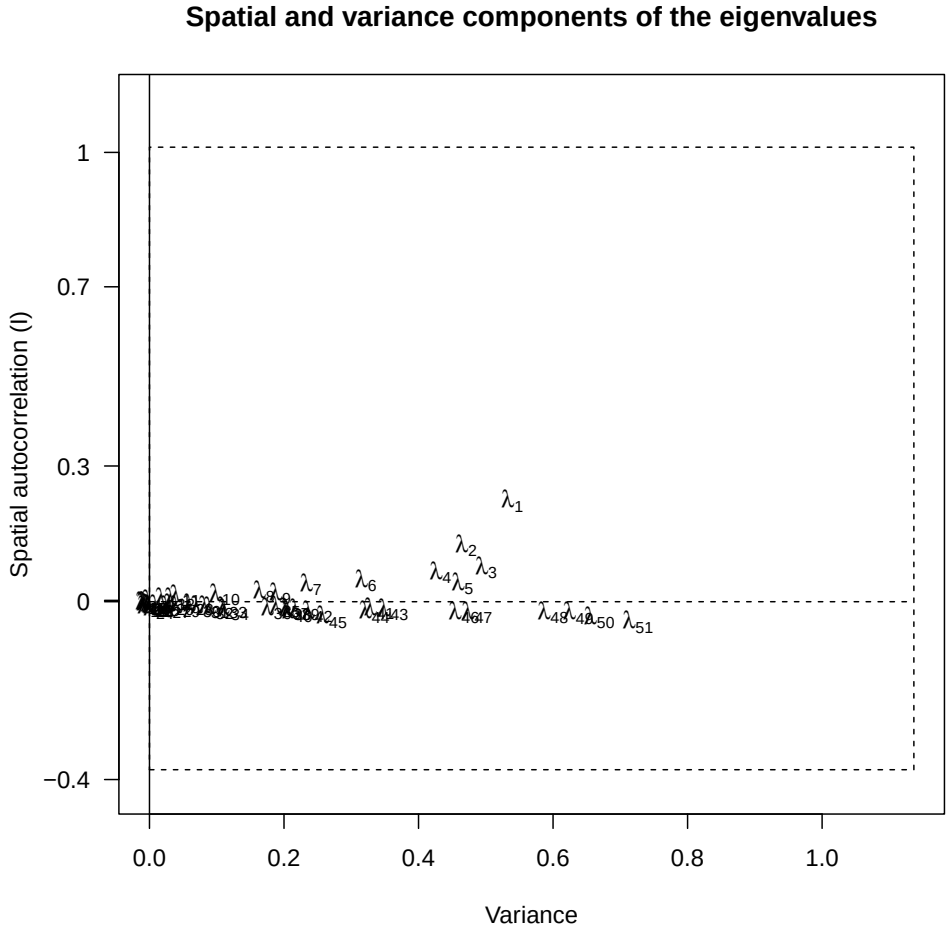
The principal components associated with the first two positive eigenvalues (in red) shall be retained. The printing of `spca` objects is more explicit than `dudi` objects, but named with the same conventions:

```
rupica.spca1

##  ########################################
##  # spatial Principal Component Analysis #
##  ########################################
## class: spca
## $call: spca.genind(obj = rupica, cn = rupica.graph, scannf = FALSE,
##     nfposi = 2, nfnega = 0)
##
## $nfposi: 2 axis-components saved
## $nfnega: 0 axis-components saved
## Positive eigenvalues: 0.03001 0.01445 0.00924 0.006851 0.004485 ...
## Negative eigenvalues: -0.008643 -0.006407 -0.004488 -0.003977 -0.003248 ...
##
##   vector length mode    content
```

```
## 1 $eig   51     numeric eigenvalues
##
##   data.frame nrow ncol content
## 1 $tab       335  55   transformed data: optionally centred / scaled
## 2 $c1        55   2    principal axes: scaled vectors of alleles loadings
## 3 $li        335  2    principal components: coordinates of entities ('scores')
## 4 $ls        335  2    lag vector of principal components
## 5 $as        2    2    pca axes onto spca axes
##
## $xy: matrix of spatial coordinates
## $lw: a list of spatial weights (class 'listw')
##
## other elements: lw
```

Unlike usual multivariate analyses, eigenvalues of sPCA are composite: they measure both the genetic diversity (variance) and the spatial structure (spatial autocorrelation measured by Moran's I). This decomposition can also be used to choose which principal component to interprete. The function `screeplot` allows to display this information graphically:

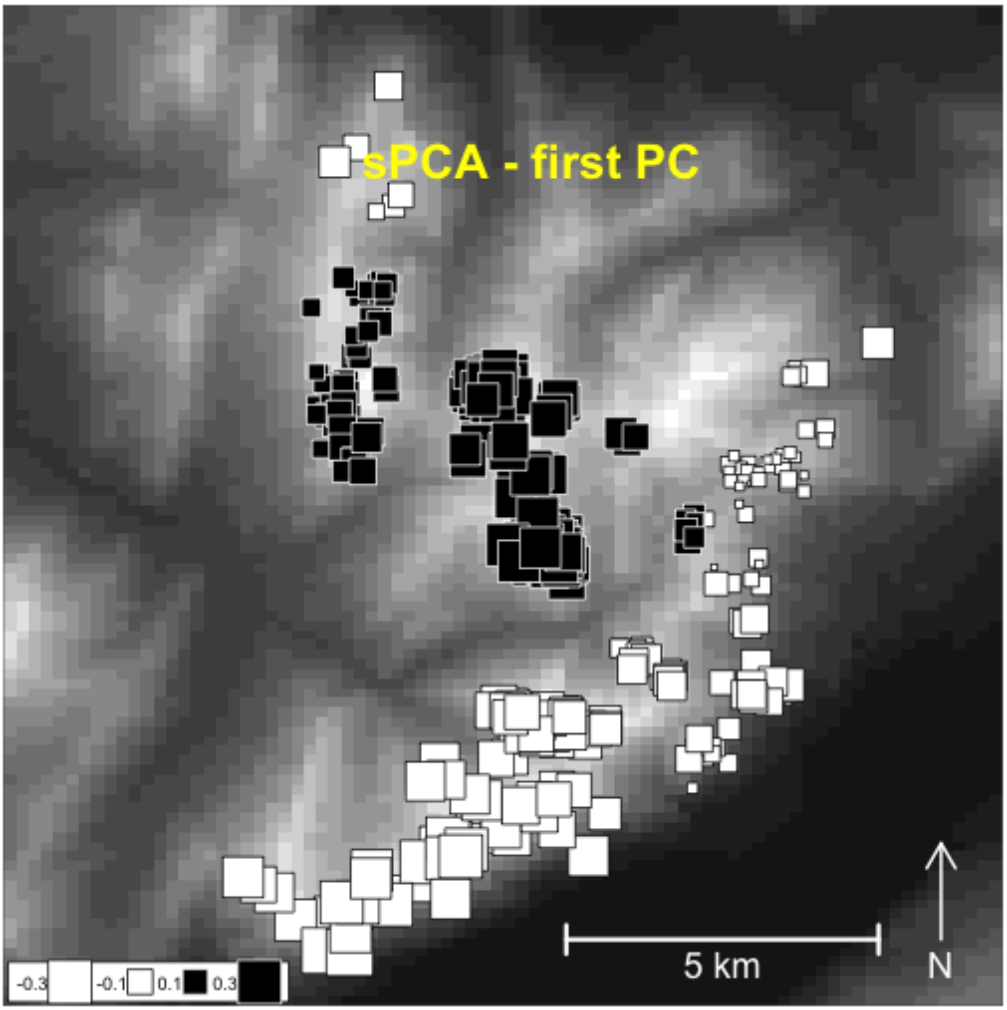
```
screeplot(rupica.spca1)
```



While λ_1 indicates with no doubt a structure, the second eigenvalue, λ_2 is less clearly distinct from the successive values. Thus, we shall keep in mind this uncertainty when interpreting the second principal component of the analysis.

We map the sPCA results using `s.value` and lagged scores (`$ls`) instead of the PC (`$li`), which are a 'denoisified' version of the PCs.

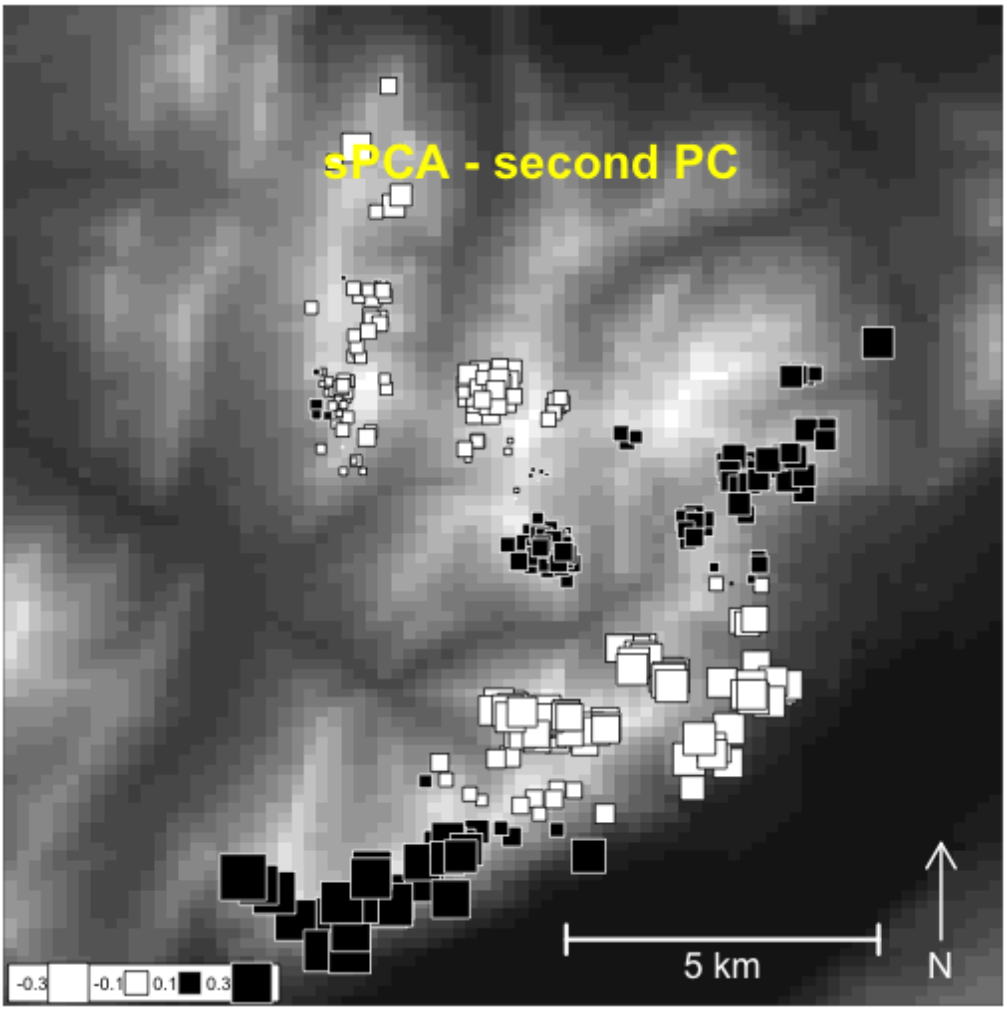
```
showBauges()
s.value(rupica$other$xy, rupica.spca1$ls[,1], add.p=TRUE, csize=0.7)
title("sPCA - first PC",col.main="yellow" ,line=-2, cex.main=2)
```



This first PC shows a remarkably clear structure opposing two high-altitude areas separated by a valley, which is thought to be an obstacle to the dispersal of Chamois (due to higher exposition to predation in low-altitude areas).

The second PC of sPCA shows an equally interesting structure:

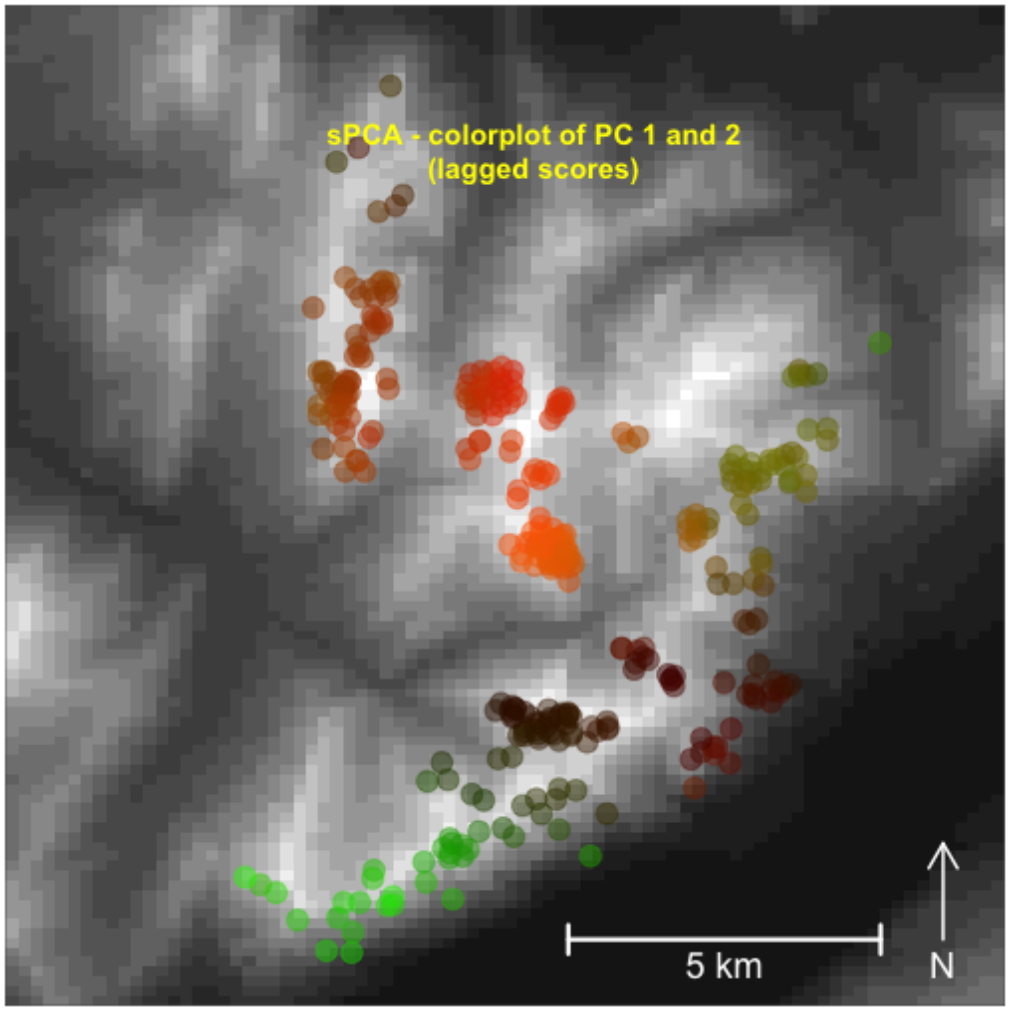
```
showBauges()
s.value(rupica$other$xy, rupica.spca1$ls[,2], add.p=TRUE, csize=0.7)
title("sPCA - second PC",col.main="yellow" ,line=-2, cex.main=2)
```



The smaller clusters appearing on this map correspond to social units identified by direct observation in the field. Therefore, this genetic structure is merely a reflect of the social behaviour of these individuals.

Both genetic structures can be represented altogether using `colorplot`. The final figure should ressemble this (although colors may change from one computer to another):

```
showBauges()
colorplot(rupica$other$xy, rupica.spca1$ls, axes=1:2, transp=TRUE, add=TRUE,
          cex=3)
title("sPCA - colorplot of PC 1 and 2\n(lagged scores)", col.main="yellow",
      line=-2, cex=2)
```


sPCA - colorplot of PC 1 and 2
(lagged scores)
5 km
N

References

[1] Jombart T, Devillard S, Dufour A-B and Pontier D (2008) Revealing cryptic spatial patterns in genetic variability by a new multivariate method. *Heredity* 101: 92-103.

[2] Jombart, T. (2008) adegenet: a R package for the multivariate analysis of genetic markers. *Bioinformatics* 24: 1403-1405.

[3] R Development Core Team (2011) R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

[4] Legendre P and Legendre L (1998) Numerical ecology. Elsevier Science B. V., Amsterdam.

[5] Moran P (1948). The interpretation of statistical maps. *Journal of the Royal Statistical Society, B* 10: 243–251.

[6] Moran, P. (1950) Notes on continuous stochastic phenomena. *Biometrika* 37: 17–23.

[7] Cliff A and Ord J (1981) *Spatial Processes. Model & Applications*. London: Pion.

[8] Menozzi P, Piazza A and Cavalli-Sforza LL (1978) Synthetic maps of human gene frequencies in Europeans. *Science* 201: 786–792.