A tutorial for spatial Analysis of Principal Components (sPCA) using *adegenet* 1.3-0

Thibaut Jombart

June 20, 2011

Abstract

This vignette provides a tutorial for the spatial analysis of principal components (sPCA, [3]) using the *adegenet* package [1] for the R software [2]. 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.

Contents

1 Introduction

This tutorial goes through the *spatial Principal Component Analysis* (sPCA, [3]), 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.

1.1 Rationale of sPCA

Mathematical notations used in this tutorial are those from [3]. The sPCA analyses a data matrix $\mathbf{X}$ 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.572-576 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, Moran's index measures only spatial structures, does not take the variability of $\mathbf{x}$ into account. 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 $\mathbf{Xv}$) decomposing C from its maximum to its minimum value. Because $C(\mathbf{Xv})$ is a product of variance and of 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 in `adegenet`.

1.2 The `spca` function

The simulated dataset used to illustrate this section has been analyzed in [3], and corresponds to Figure 2A of the article. In `adegenet`, the matrix of alleles frequencies previously denoted $\mathbf{X}$ exactly corresponds to the `@tab` slot of `genind` or `genpop` objects:

```
> library(adegenet)
> data(spcaIllus)
> obj <- spcaIllus$dat2A
> obj


   ####################
   ### Genind object ###
   ####################
- genotypes of individuals -

S4 class:  genind
@call: old2new(object = obj)

@tab:  80 x 192 matrix of genotypes

@ind.names: vector of  80 individual names
@loc.names: vector of  20 locus names
@loc.nall: number of alleles per locus
@loc.fac: locus factor for the  192 columns of @tab
@all.names: list of  20 components yielding allele names for each locus
@ploidy:  2
@type:  codom

Optionnal contents:
@pop:  factor giving the population of each individual
@pop.names:  factor giving the population of each individual

@other: a list containing: xy


> head(truenames(obj[loc = "L01"])$tab)


     L01.1 L01.2 L01.3 L01.4 L01.5 L01.6 L01.7 L01.8 L01.9
0035     0     0   0.0     0   0.5   0.5     0   0.0   0.0
0352     0     0   0.5     0   0.5   0.0     0   0.0   0.0
0423     0     0   0.0     0   0.5   0.0     0   0.0   0.5
0289     0     0   0.0     0   0.0   0.5     0   0.0   0.5
0487     0     0   0.0     0   0.0   0.5     0   0.5   0.0
0053     0     0   0.0     0   0.5   0.5     0   0.0   0.0
```

The object `obj` is a `genind` object; note that here, we only displayed the table for the first locus (`loc="L01"`).

The function performing the sPCA is `spca`; it accepts a bunch of arguments, but only the first two are mandatory to perform the analysis (see `?spca` for further information):

```
> args(spca)
```

```
function (obj, xy = NULL, cn = NULL, matWeight = NULL, scale = FALSE,
    scale.method = c("sigma", "binom"), 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`/`genpop` 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 not require a `xy` argument. Basically, spatial information could be stored in any form and with any name in the `@other` slot, but the `spca` function would not recognize it directly. Note that `obj` already contains spatial coordinates at the appropriate place. Hence, the following uses are valid (`ask` and `scannf` are set to FALSE to avoid interactivity):

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


deldir 0.0-13

     Please note: The process for determining duplicated points
     has changed from that used in version 0.0-9 (and previously).


> mySpca2 <- spca(obj, xy = other(obj)$xy, ask = FALSE, type = 1,
+     scannf = FALSE)
> all.equal(mySpca, mySpca2)


[1] "Component 8: target, current do not match when deparsed"


> names(mySpca)[8]


[1] "call"
```

Both objects are the same: they only differ by their call.

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 does not directly use a matrix of spatial weightings, but a connection network with the class `nb` or a list of spatial weights of class `listw`, which are both implemented in the `spdep` package. The function `chooseCN` is a wrapper for different functions spread in 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 from their distances (`type=5`), considering as neighbours two entities whose distance between 0 (`d1=0`) and 2(`d2=2`).

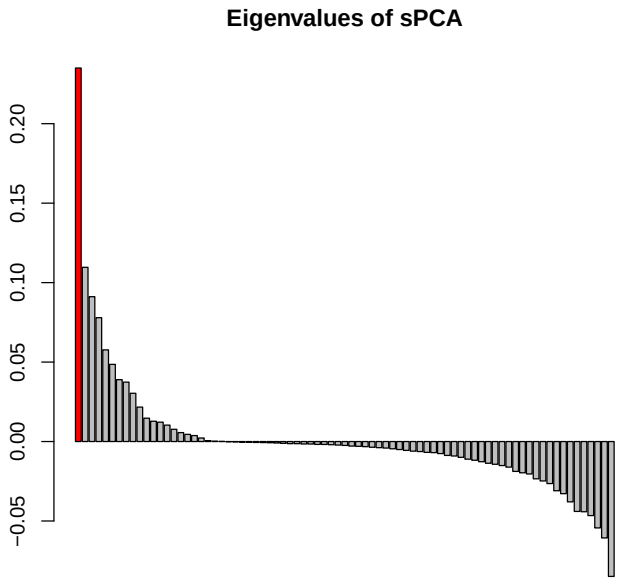
Another possibility is of course to provide directly a connection network (`nb` object) or a list of spatial weights (`listw` object) to the `spca` function; this can be done via the `cn` argument. For instance:

```
> 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.

After providing a genetic dataset along with a spatial information, the `spca` function displays a barplot of eigenvalues and asks for a number of positive axes ('first number of axes') and negative axes ('second number of axes') to be retained (unless `scannf` is set to `FALSE`). 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 these information have 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 as connection network:

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

```
[1] "spca"
```

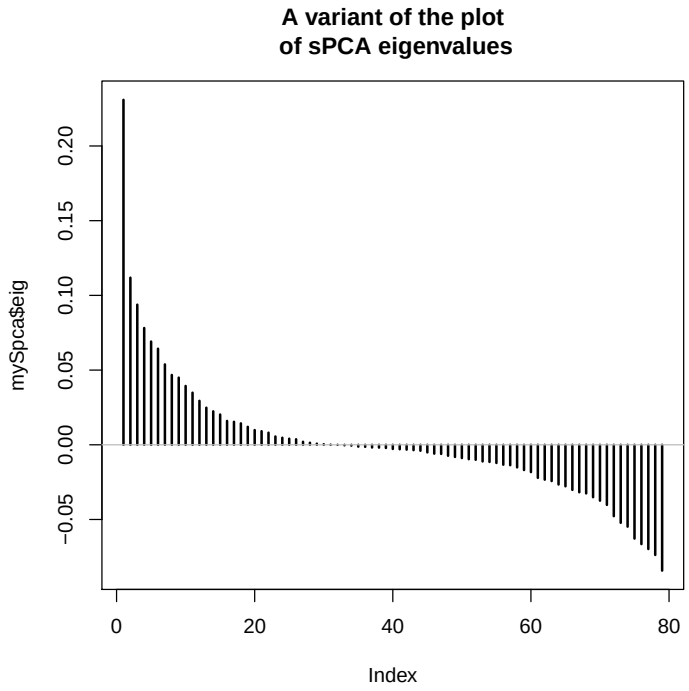
```
> mySpca
```

```
        ########################################
        # spatial Principal Component Analysis #
        ########################################
class: spca
$call: spca(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 $c1        192  1    principal axes: scaled vectors of alleles loadings
2 $li        80   1    principal components: coordinates of entities ('scores')
3 $ls        80   1    lag vector of principal components
4 $as        2    1    pca axes onto spca axes

$xy: matrix of spatial coordinates
$lw: a list of spatial weights (class 'listw')

other elements: NULL
```

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 [3] 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


> plot(mySpca$eig, type = "h", lwd = 2, main = "A variant of the plot\n of sPCA eigenvalues")
> abline(h = 0, col = "grey")
```



The axes of the analysis, denoted **v** in eq. (4) [3] are stored as columns inside the `$c1` component. Each columns contains loadings for all the alleles:

```
> mySpca$c1
```

```
               Axis 1
L01.1    1.268838e-02
L01.2    2.220446e-16
L01.3   -1.119979e-01
L01.4   -4.440892e-16
L01.5   -2.766095e-02
L01.6   -4.477031e-02
L01.7   -8.673617e-19
L01.8    7.555869e-02
L01.9    9.618208e-02
L02.01  -3.689948e-02
L02.02   8.065343e-03
L02.03   3.792050e-02
L02.04   6.383679e-03
L02.05   8.818737e-02
L02.06  -5.804721e-02
L02.07  -3.191157e-02
L02.08  -1.033976e-25
L02.09  -7.866429e-02
L02.10   0.000000e+00
L02.11   6.496566e-02
L03.1   -1.009742e-28
L03.2    2.084734e-02
L03.3    5.405746e-02
L03.4    1.226999e-01
L03.5   -1.372961e-01
L03.6   -1.540744e-33
L03.7   -3.362424e-02
L03.8   -2.668446e-02
L04.1   -1.362459e-01
```

```
L04.2    1.443615e-01
L04.3    0.000000e+00
L04.4    0.000000e+00
L04.5   -5.741018e-02
L04.6   -1.147944e-41
L04.7    1.728662e-03
L04.8    4.756588e-02
L05.1   -5.605194e-45
L05.2   -3.503246e-46
L05.3   -5.665231e-02
L05.4    0.000000e+00
L05.5    4.276424e-50
L05.6   -2.672765e-51
L05.7    8.350792e-03
L05.8    1.339803e-01
L05.9   -8.567875e-02
L06.01  -3.102925e-02
L06.02   4.527011e-02
L06.03  -1.991365e-59
L06.04  -3.111508e-61
L06.05  -9.921947e-03
L06.06  -3.192068e-02
L06.07   9.464883e-02
L06.08  -9.153327e-02
L06.09  -4.446149e-03
L06.10  -1.475188e-02
L06.11   4.368425e-02
L07.1   -5.912766e-02
L07.2    0.000000e+00
L07.3    1.087945e-01
L07.4   -3.516619e-02
L07.5   -6.995648e-02
L07.6    5.881782e-03
L07.7    1.301858e-02
L07.8   -2.635286e-02
L07.9    6.290836e-02
L08.01   1.130714e-02
L08.02   4.021857e-02
L08.03   1.994916e-02
L08.04  -4.794037e-94
L08.05   3.523383e-04
L08.06  -1.744903e-01
L08.07   1.742903e-01
L08.08 -1.785918e-102
L08.09   0.000000e+00
L08.10 -5.450189e-107
L08.11  -7.162714e-02
L09.01   1.184934e-01
L09.02  -2.709607e-03
L09.03   5.920972e-03
L09.04  -4.335124e-02
L09.05  -7.487314e-02
L09.06  -8.335186e-02
L09.07   1.497592e-02
L09.08   0.000000e+00
L09.09   1.113123e-02
L09.10   0.000000e+00
L09.11   5.376433e-02
L10.1    2.504824e-02
L10.2    7.072426e-02
L10.3   -3.036288e-02
L10.4    7.391488e-02
L10.5   -1.288548e-01
L10.6    0.000000e+00
L10.7    2.504824e-02
L10.8   -3.551790e-02
L11.1    0.000000e+00
L11.2    2.036407e-02
L11.3    0.000000e+00
L11.4   -2.472112e-01
L11.5    1.122655e-01
L11.6    1.145816e-01
L12.1    0.000000e+00
L12.2   -2.658753e-02
L12.3    0.000000e+00
L12.4   -1.788594e-01
```

```
L12.5    1.340382e-03
L12.6    0.000000e+00
L12.7    7.896368e-02
L12.8    1.251428e-01
L13.01   0.000000e+00
L13.02   1.332178e-02
L13.03  -3.249165e-03
L13.04  -1.339280e-02
L13.05  -1.460385e-01
L13.06   9.330816e-02
L13.07   0.000000e+00
L13.08   3.120220e-02
L13.09   9.509832e-04
L13.10   2.389730e-02
L14.01   0.000000e+00
L14.02   6.235773e-02
L14.03  -7.948718e-02
L14.04   1.113123e-02
L14.05  -8.394201e-02
L14.06  -3.414988e-02
L14.07  -4.773145e-02
L14.08   1.807464e-02
L14.09   6.383679e-03
L14.10  -1.995468e-02
L14.11   1.673179e-01
L15.01   3.194486e-02
L15.02   8.033966e-03
L15.03  -8.388260e-02
L15.04   0.000000e+00
L15.05   4.768429e-02
L15.06   1.780219e-02
L15.07   5.466061e-02
L15.08   0.000000e+00
L15.09  -8.106139e-02
L15.10  -4.255964e-02
L15.11   0.000000e+00
L15.12   2.374409e-02
L15.13   0.000000e+00
L15.14   2.363363e-02
L16.01  -1.095284e-02
L16.02  -1.695265e-01
L16.03   2.243837e-02
L16.04  -3.697041e-02
L16.05   4.411850e-02
L16.06  -6.477905e-02
L16.07  -8.361412e-03
L16.08  -2.803524e-04
L16.09   2.891119e-02
L16.10   1.675135e-01
L16.11   2.788895e-02
L17.1    1.357193e-01
L17.2   -2.406925e-01
L17.3    8.463989e-05
L17.4    8.132827e-02
L17.5    3.841352e-02
L17.6    1.167365e-01
L17.7   -1.315896e-01
L18.01   0.000000e+00
L18.02   0.000000e+00
L18.03  -3.108654e-02
L18.04  -7.231939e-02
L18.05   1.113227e-01
L18.06  -1.097742e-01
L18.07   5.267695e-02
L18.08  -3.711225e-03
L18.09   2.712486e-02
L18.10   2.576680e-02
L19.01   0.000000e+00
L19.02  -3.802937e-02
L19.03   0.000000e+00
L19.04  -1.411136e-01
L19.05   8.463254e-02
L19.06   9.207072e-03
L19.07  -4.750854e-02
L19.08  -2.922071e-02
L19.09   0.000000e+00
```

```
L19.10  -4.251438e-02
L19.11   5.197636e-02
L19.12   1.525706e-01
L20.1   -2.598833e-02
L20.2    1.888407e-02
L20.3    2.871585e-01
L20.4    1.485180e-02
L20.5   -1.500353e-02
L20.6    1.659481e-02
L20.7   -1.426074e-01
L20.8   -1.538899e-01
```

```
> head(mySpca$c1)
```

```
             Axis 1
L01.1  1.268838e-02
L01.2  2.220446e-16
L01.3 -1.119979e-01
L01.4 -4.440892e-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 ordinary, '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, this value of the projection simply is the correlation between both axes. This information is stored inside the `$as` component:

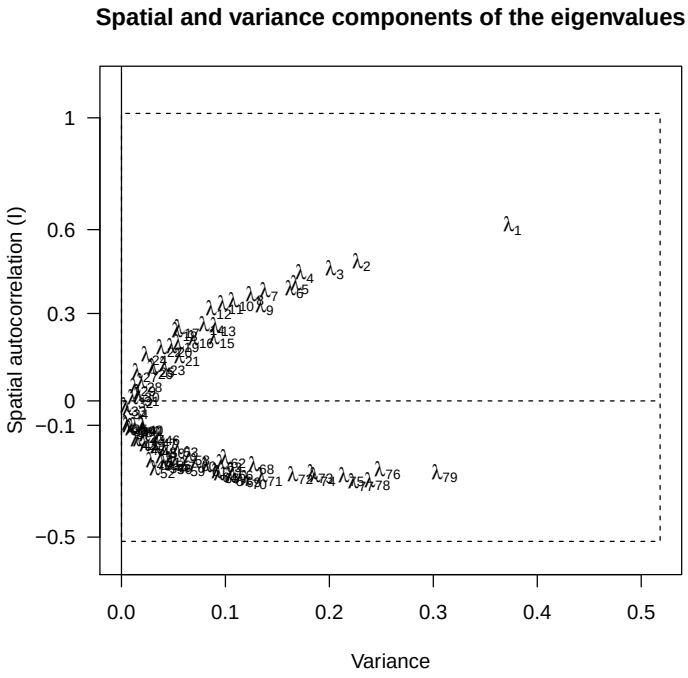
```
> mySpca$as
```

```
              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)
```

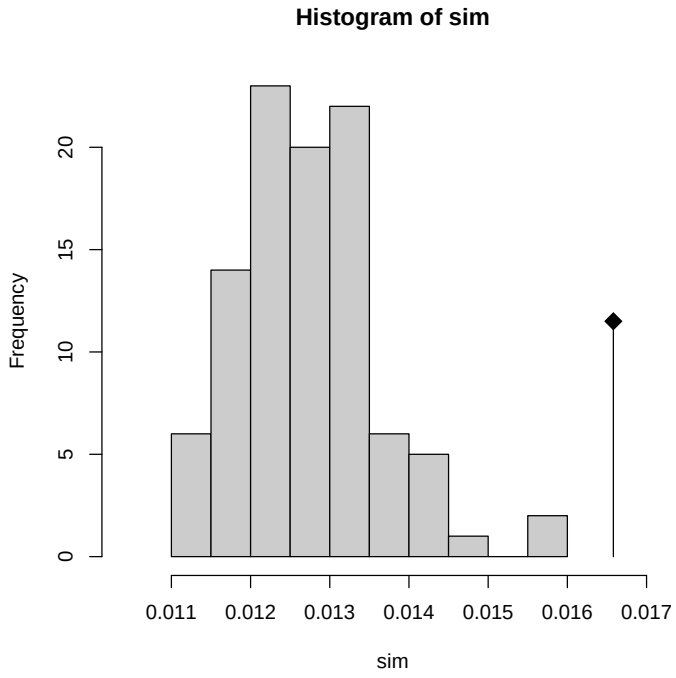


The resulting figure represents eigenvalues of sPCA (denoted λ_i with $i = 1, \ldots, n-1$, where λ_1 is the strongest global eigenvalue, and λ_{n-1} is the strongest negative eigenvalue) according the their variance and Moran's I components. These eigenvalues are contained inside a rectangle indicated 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 [3] 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
5.500655e+00 1.273582e-02 4.886649e-07


> plot(myGtest)
```

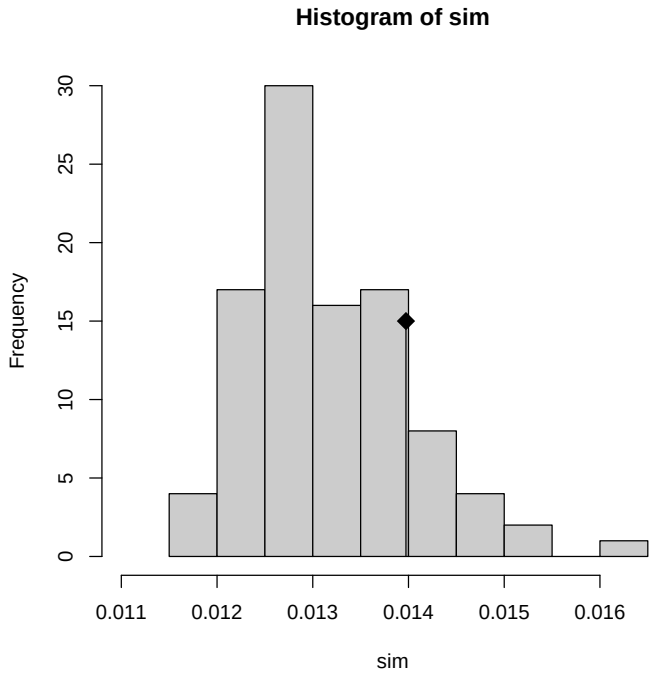


The produced object is a `randtest` object (see `?randtest`), which is the class of objects for Monte-Carlo tests in the *ade4* package. As shown, such object can be plotted using a `plot` function: the resulting figure shows an histogram of permuted test statistics and indicates the observed statistics 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 alternative hypothesis. Note that because 99 permutations were used, the p-value cannot be lower than 0.01. In practice, more permutations should be used (like 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:

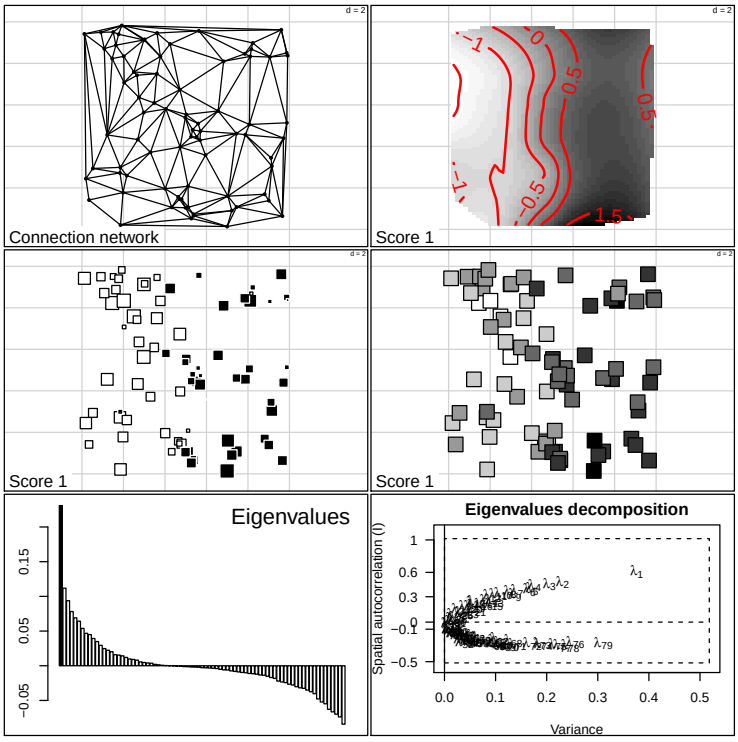
```
> myLtest <- local.rtest(obj$tab, mySpca$lw, nperm = 99)
> myLtest
```

```
Monte-Carlo test
Call: local.rtest(X = obj$tab, listw = mySpca$lw, nperm = 99)

Observation: 0.01397349

Based on 99 replicates
Simulated p-value: 0.17
Alternative hypothesis: greater

     Std.Obs  Expectation     Variance
9.766686e-01 1.321484e-02 6.033766e-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)


Spatial Point Pattern Analysis Code in S-Plus

 Version 2 - Spatial and Space-Time analysis
```

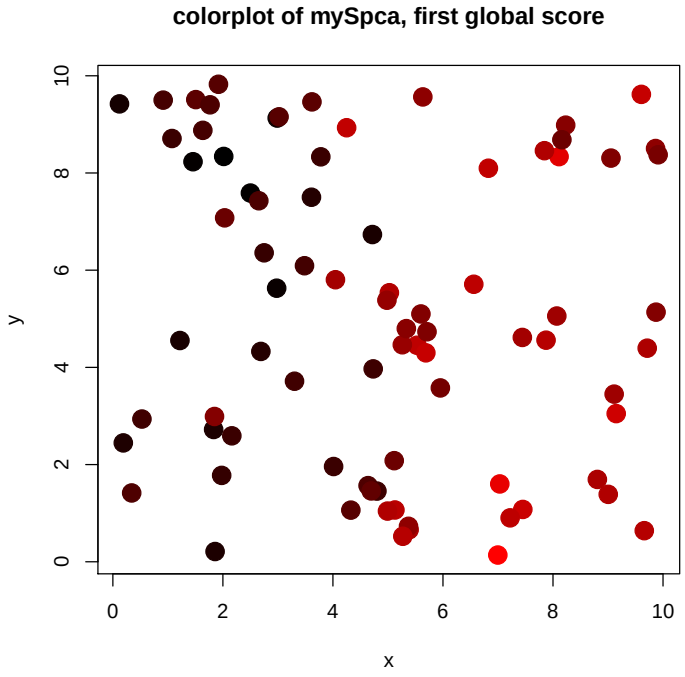


This figure shows different information, that we detail from the top to bottom and from left to right. The first plot shows the connection network that was used to define spatial weightings. The second, third, and fourth plots are different representations of a score of entities in space, the first global 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.

Another way of representing a score of sPCA is using the `colorplot` function. This function can show 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`:

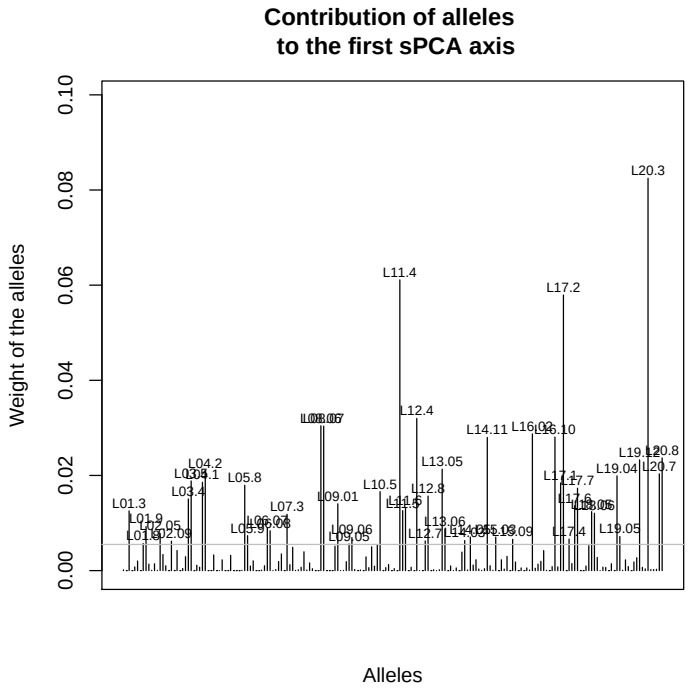
```
> colorplot(mySpca, cex = 3, main = "colorplot of mySpca, first global score")
```



See `example(colorplot)` and `example(spca)` for more examples of applications of colorplot to represent sPCA scores.

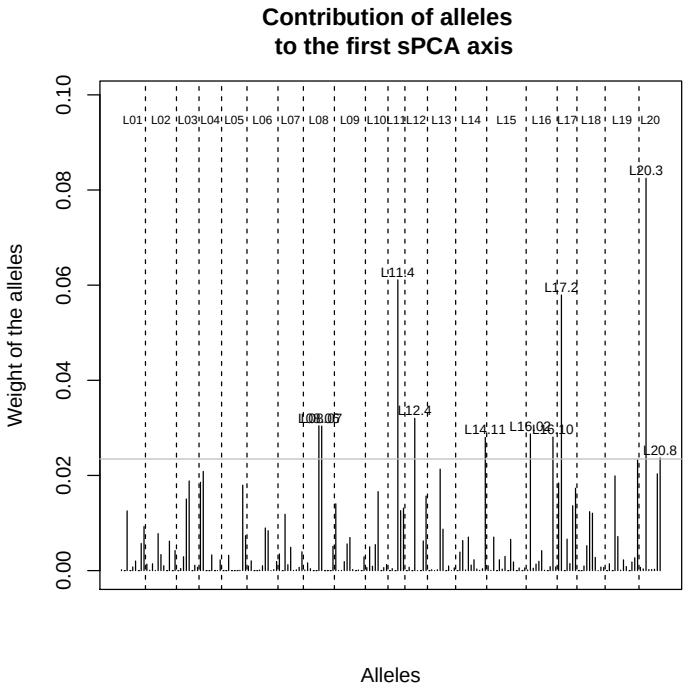
So far, we assessed the spatial genetic structures existing in the data. We learned that a global structure existed, and we observed that it consisted in two east-west genetic clusters. Now, we may like to know how each allele contributes to a given score. To quantify such contribution, only the absolute value of loadings for a given structure is meaningful. However, it is more relevant to consider squared loadings, as their sum is always constrained to be unit (because $\|\mathbf{v}\|^2 = 1$). We can look for the alleles contributing most to 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")
```



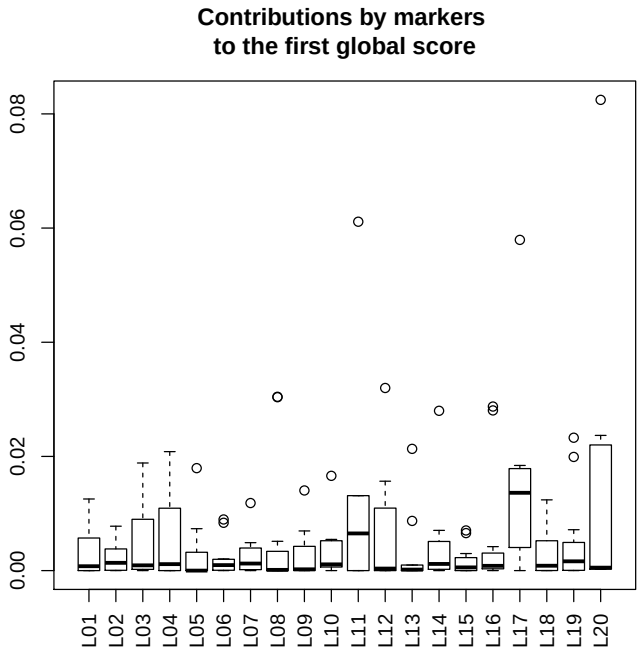
See `?loadingplot` for more information about this function, in particular for the definition of the threshold value above which alleles are annotated. Note that it is possible to also separate the alleles by markers, using the `fac` argument, to assess if all markers have comparable contributions to 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"
 [9] "L20.3"  "L20.8"

$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
0.03044687 0.03037709 0.06111338 0.03199067 0.02799529 0.02873923 0.02806079
     L17.2      L20.3      L20.8
0.05793290 0.08246000 0.02368209
```



But to assess the average contribution of each marker, a traditional boxplot remains a better tool:

```
> boxplot(myLoadings ~ obj$loc.fac, las = 3, main = "Contributions by markers \nto the first global score")
```



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 in this exercise.

2.1 An overview of the data

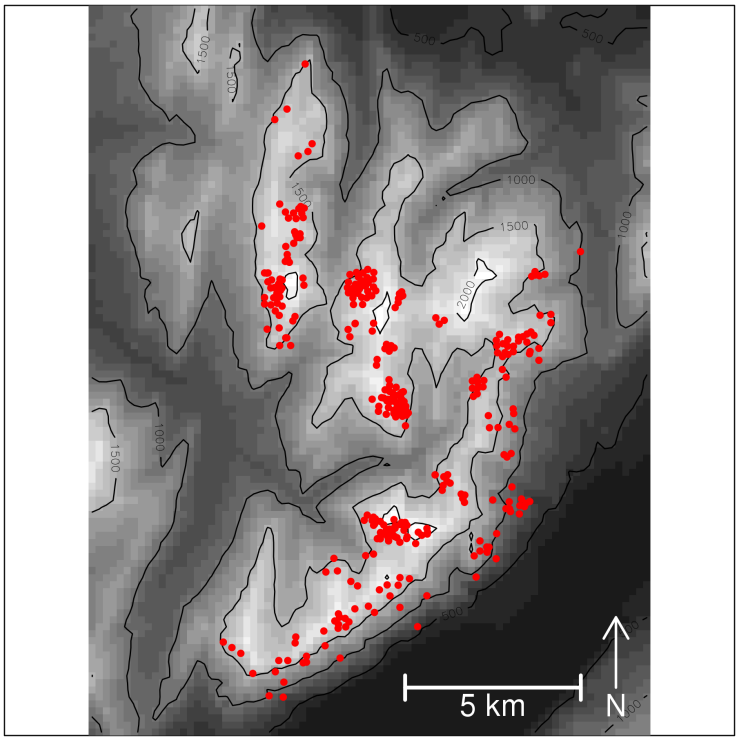
We first load the data:

```
> data(rupica)
> rupica
```

```
   ####################
   ### Genind object ###
   ####################
- genotypes of individuals -

S4 class:  genind
@call: NULL

@tab:  335 x 55 matrix of genotypes

@ind.names: vector of  335 individual names
@loc.names: vector of  9 locus names
@loc.nall: number of alleles per locus
@loc.fac: locus factor for the  55 columns of @tab
@all.names: list of  9 components yielding allele names for each locus
@ploidy:  2
@type:  codom

Optionnal contents:
@pop:  - empty -
@pop.names:  - empty -

@other: a list containing: xy  mnt  showBauges
```

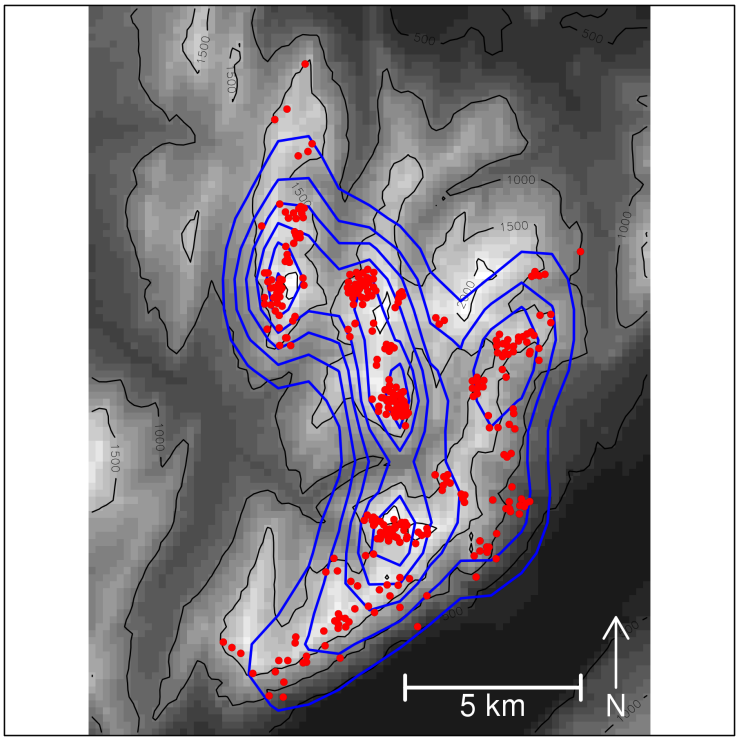
`rupica` is a typical `genind` object, which is the class of objects storing genotypes (as opposed to population data) in *adegenet*. `rupica` also contains topographic information about the sampled area, which can be displayed by calling `rupica$other$showBauges`. For instance, the spatial distribution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

```
> rupica$other$showBauges()


Be careful: it is now recommended to use the newpackages adehabitatMA, adehabitatLT, adehabitatHR, and adehabitat
These 4 packages are intended to become the future of adehabitat.
The "classical" version of adehabitat will still be maintained for some
time, but no new method will be added to the package.


> 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)
```



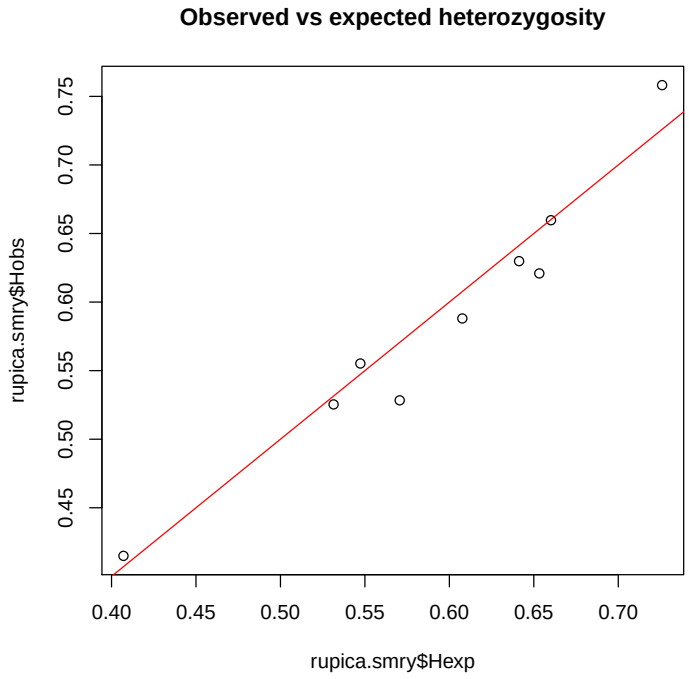
Is geographical clustering strong enough to assign safely each individual to a group? Accordingly, shall we analyse these data at individual or group level?

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)


 # Total number of genotypes:  335

 # Population sample sizes:

335

 # Number of alleles per locus:
L1 L2 L3 L4 L5 L6 L7 L8 L9
 7 10  7  6  5  5  6  4  5

 # Number of alleles per population:
 1
55

 # Percentage of missing data:
[1] 0

 # Observed heterozygosity:
```

```
       L1        L2        L3        L4        L5        L6        L7        L8
0.5880597 0.6208955 0.5253731 0.7582090 0.6597015 0.5283582 0.6298507 0.5552239
       L9
0.4149254

 # Expected heterozygosity:
       L1        L2        L3        L4        L5        L6        L7        L8
0.6076769 0.6532517 0.5314591 0.7259657 0.6601604 0.5706082 0.6412742 0.5473112
       L9
0.4070709


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

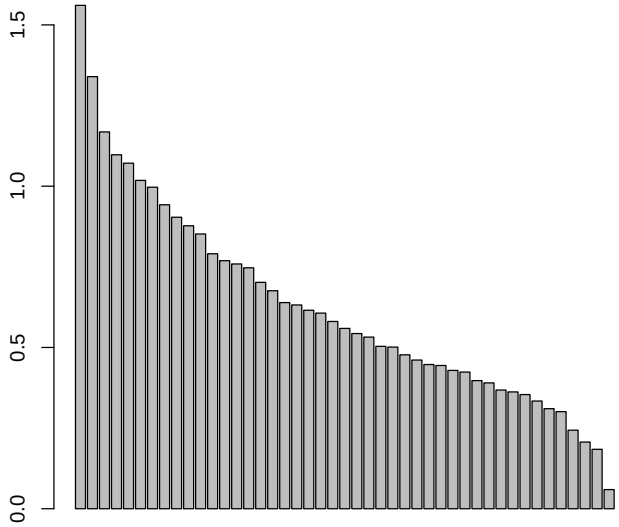


The red line indicate identity between both quantities. What can we say about heterozygosity in this population? How can this be tested? The result below can be reproduced using a standard testing procedure:

```
> 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 = 0.9461, df = 8, p-value = 0.3718
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.01025068  0.02451318
sample estimates:
mean of the differences
            0.007131251
```

We can seek a global picture of the genetic diversity among genotypes using a Principal Component Analysis (PCA, [**?**, **?**], `dudi.pca` in `ade4` package). The analysis is performed on a table of standardised alleles frequencies, obtained by `scaleGen` (use the binomial scaling option). Remember to disable the scaling option when performing the PCA. The function `dudi.pca` displays a barplot of eigenvalues and asks for a number of retained principal components:

```
> rupica.X <- scaleGen(rupica, method = "binom")
> rupica.pca1 <- dudi.pca(rupica.X, cent = FALSE, scale = FALSE,
+     scannf = FALSE, nf = 2)
> barplot(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, center = FALSE, scale = FALSE, scannf = FALSE,
    nf = 2)

$nf: 2 axis-components saved
```

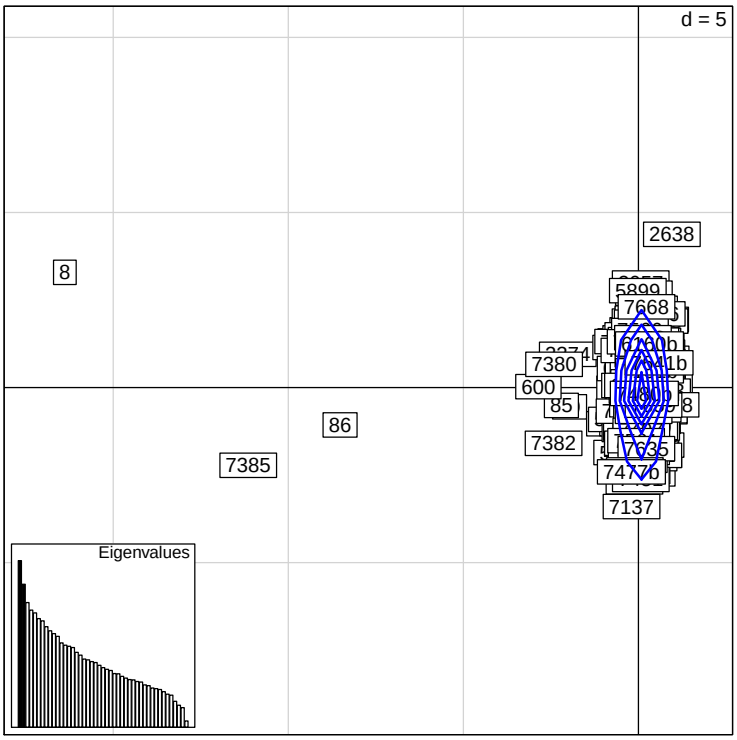
```
$rank: 45
eigen values: 1.561 1.34 1.168 1.097 1.071 ...
  vector length mode    content
1 $cw    55     numeric column weights
2 $lw    335    numeric row weights
3 $eig   45     numeric eigen values

  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). Verify that here, each eigenvalue is the variance of the corresponding PC.

An abrupt decrease in eigenvalues is likely to indicate the boundary between true patterns and non-interpretable structures. In this case, how many PCs would you interprete?

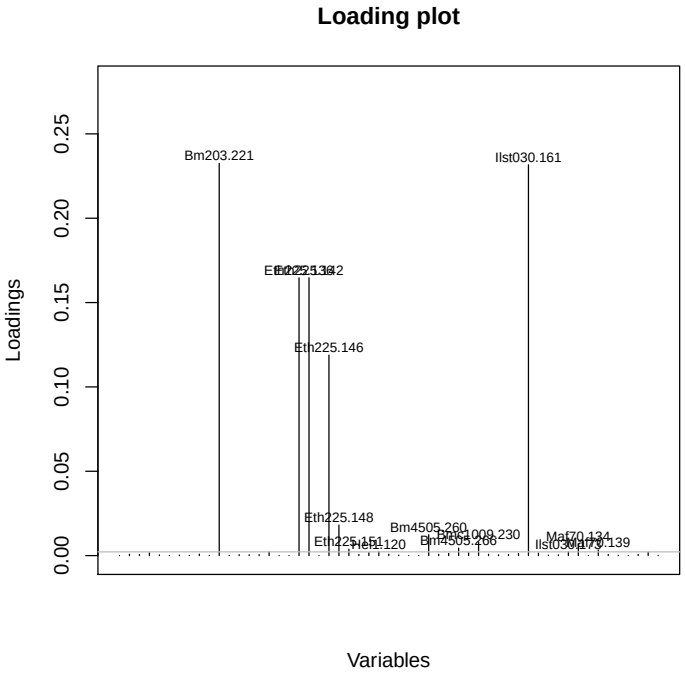
Use `s.label` to display to two first components of the analysis. Then, use a kernel density (`s.kde2d`) for a better assessment of the distribution of the genotypes onto the principal axes:

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



What can we say about the genetic diversity among these genotypes as inferred by PCA? The function `loadingplot` allows to visualize the contribution of each allele, expressed as squared loadings, for a given principal component. Using this function, reproduce this figure:

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



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

```
> X <- truenames(rupica)
> class(X)

[1] "matrix"

> dim(X)

[1] 335  55

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

bm203.221
                  0 0.00597014925373134                   0.5
                330                   1                     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?

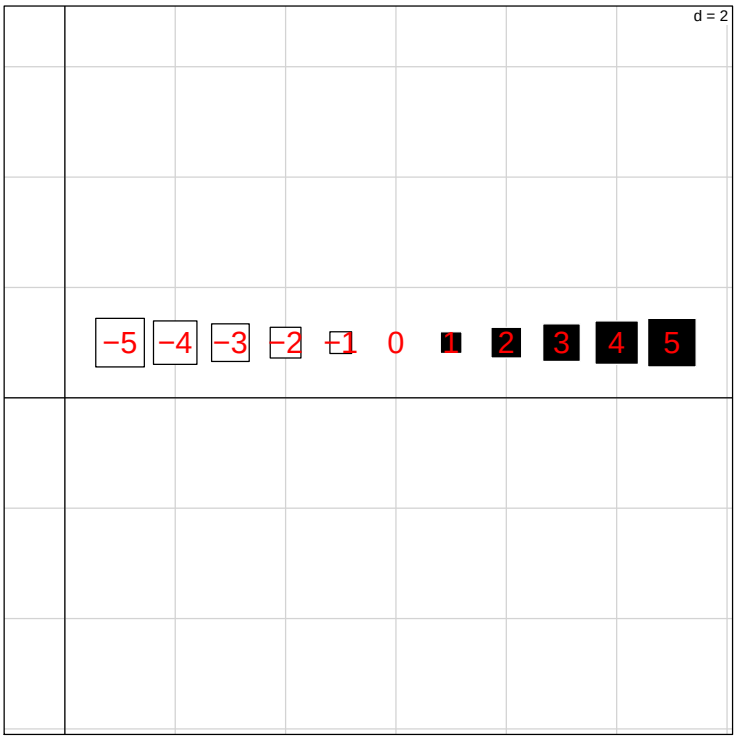
```
> rownames(X)[bm203.221 == 0.5]

   001    019    029    276
   "8"   "86"  "600" "7385"
```

Conclusion?

2.3 Mapping and testing PCA results

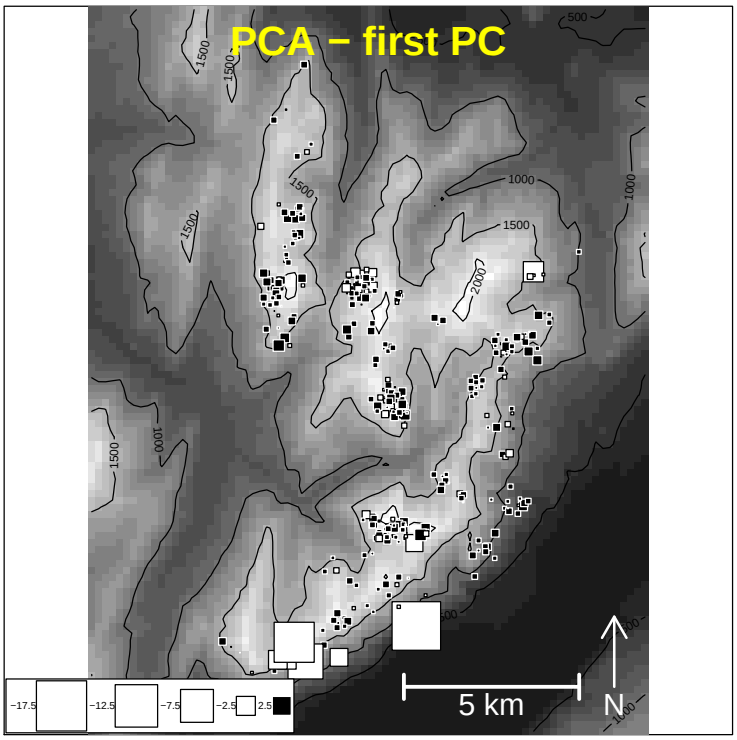
A frequent practice in spatial genetics is mapping the first principal components (PCs) onto the geographic space. The function `s.value` is well-suited to do so, using black and white squares of variable size for positive and negative values. To give a legend for this type of representation:

```
> 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)
```

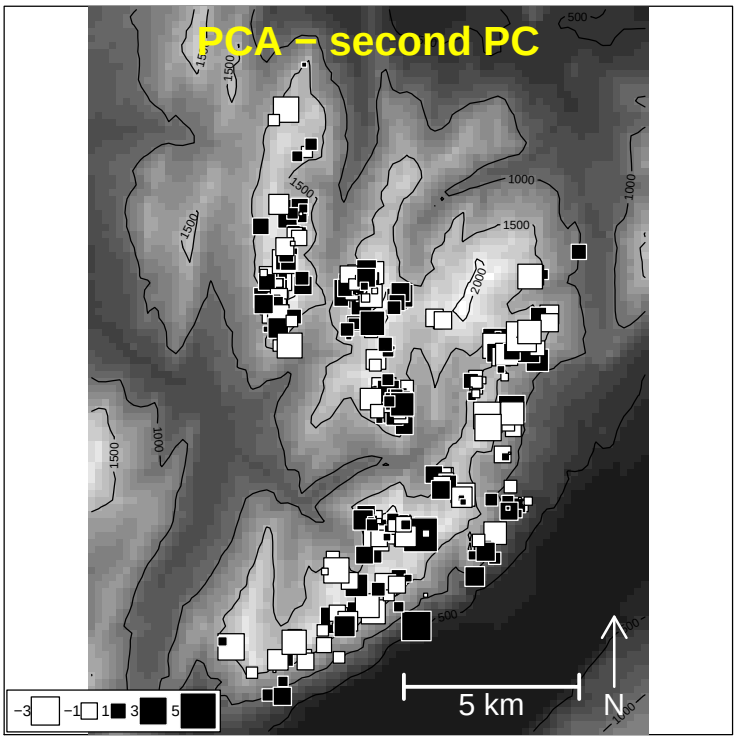


Apply this graphical representation to the first two PCs of the PCA:

```
> 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)
```



What can we say about spatial genetic structure as inferred by PCA? This visual assessment can be complemented by testing the spatial autocorrelation in the first PCs of PCA. This can be achieved using Moran's I test. Use the function `moran.mc` in the package *spdep* to perform these tests. You will need first 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. Home ranges will be modelled as disks with a radius of 1150m. Use `chooseCN` to create a connection network based on distance range ("neighbourhood by distance"). What threshold distance do you choose for individuals to be considered as neighbours?

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

The connection network should ressemble this:

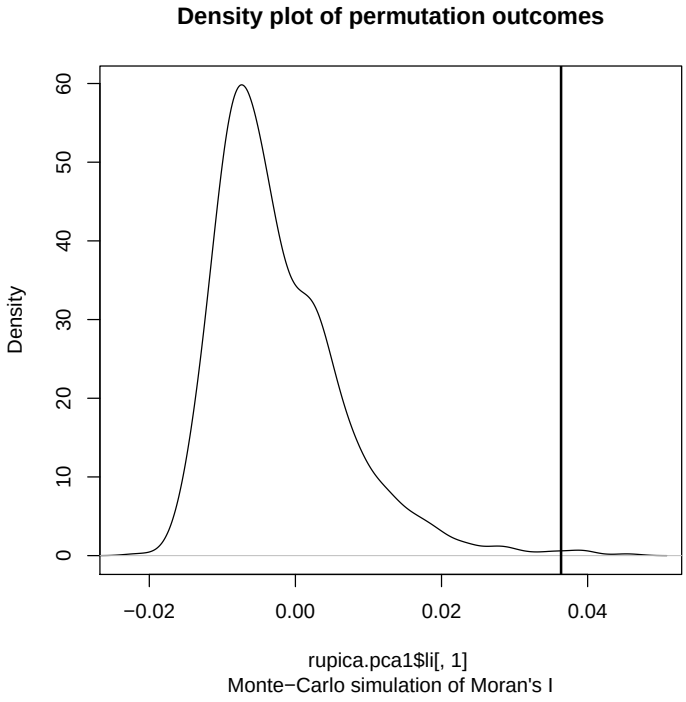
```
> rupica.graph
```

```
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")
```

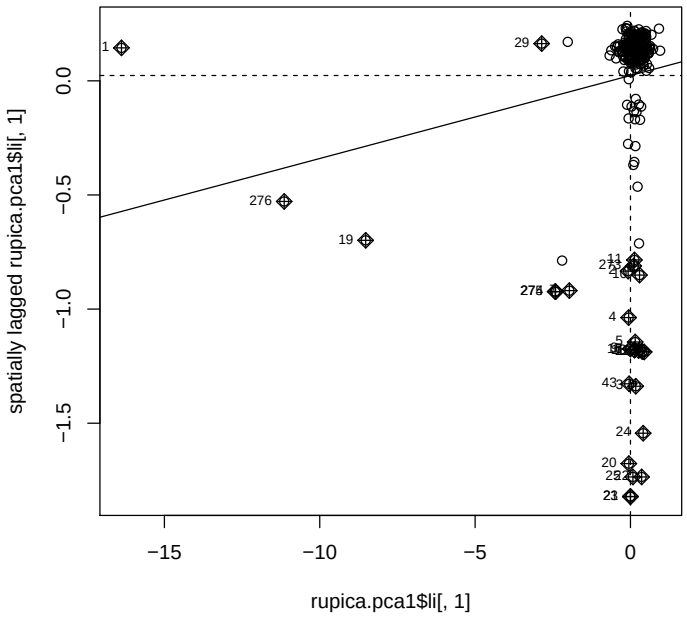
Perform Moran's test for the first two PCs, and plot the results. The first test should be significant:

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



Compare this result to the mapping of the first PC of PCA. What is wrong? When a test gives unexpected results, it is worth looking into the data in more details. Moran's plot (`moran.plot`) plots the tested variable against its lagged vector. Use it on the first PC:

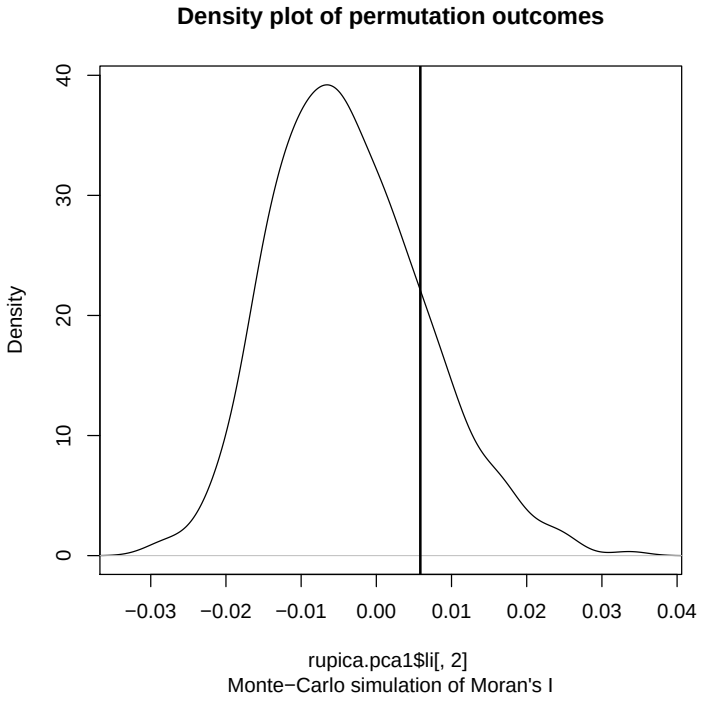
```
> moran.plot(rupica.pca1$li[, 1], rupica.graph)
```



Actual positive autocorrelation corresponds to a positive correlation between a variable and its lag vector. Is it the case here? How can we explain that Moran's test was significant?

Repeat these analyses for the second PC. What are your conclusions?

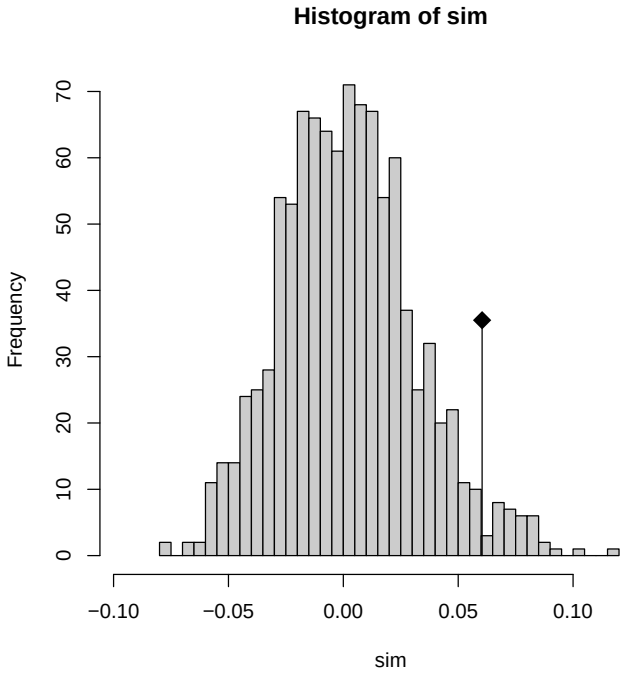
```
> pc2.mctest <- moran.mc(rupica.pca1$li[, 2], rupica.graph, 999)
> plot(pc2.mctest)
```



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)
```



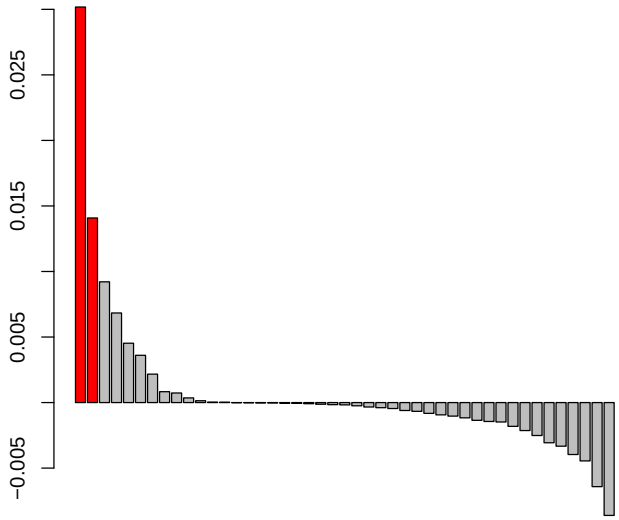
What is your conclusion? Shall we be looking for spatial structures? If so, how can we explain that PCA did not reveal them? Does the Mantel correlogram (`mantel.correlog` in *vegan* package) bring any help solving the problem?

2.5 spatial Principal Component Analysis

The spatial Principal Component Analysis (sPCA, function `spca` [3]) has been especially developed to investigate hidden or non-obvious spatial genetic patterns. Like Moran's I test, sPCA first requires the spatial proximities between genotypes to be modeled. You will reuse the connection network defined previously using `chooseCN`, and pass it as the 'cn' argument of the function `spca`.

Read the documentation of `spca`, and apply the function to the dataset `rupica`. The function will display a barplot of eigenvalues:

```
> 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)))
```



This figure illustrates the fundamental difference between PCA and sPCA. Like `dudi.pca`, `spca` displays a barplot of eigenvalues, but unlike in PCA, eigenvalues of sPCA can also be negative. This is because the criterion optimized by the analysis can have positive and negative values, corresponding respectively to positive and negative autocorrelation. Positive spatial autocorrelation correspond to greater genetic similarity between geographically closer individuals. Conversely, negative spatial autocorrelation corresponds to greater dissimilarity between neighbours. The spatial autocorrelation of a variable is measured by Moran's I, and interpreted as follows:

- $I_0 = -1/(n-1)$: no spatial autocorrelation (x is randomly distributed across space)
- $I > I_0$: positive spatial autocorrelation
- $I < I_0$: negative spatial autocorrelation

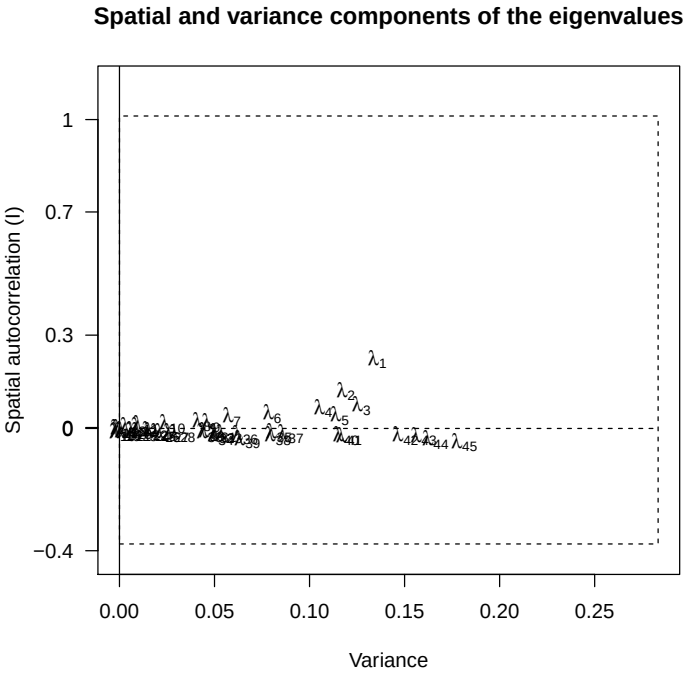
Principal components of PCA ensure that (ϕ referring to one PC) $var(\phi)$ is maximum. By contrast, sPCA provides PC which decompose the quantity $var(\phi)I(\phi)$. In other words, PCA focuses on variability only, while sPCA is a compromise between variability ($var(\phi)$) and spatial structure ($I(\phi)$).

In this case, only the principal components associated with the two first 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(obj = rupica, cn = rupica.graph, scannf = FALSE, nfposi = 2,
    nfnega = 0)

$nfposi: 2 axis-components saved
$nfnega: 0 axis-components saved
Positive eigenvalues: 0.03018 0.01408 0.009211 0.006835 0.004529 ...
Negative eigenvalues: -0.008611 -0.006414 -0.004451 -0.003963 -0.003329 ...

  vector length mode    content
1 $eig   45     numeric eigenvalues

  data.frame nrow ncol content
1 $c1        55   2    principal axes: scaled vectors of alleles loadings
2 $li        335  2    principal components: coordinates of entities ('scores')
3 $ls        335  2    lag vector of principal components
4 $as        2    2    pca axes onto spca axes

$xy: matrix of spatial coordinates
$lw: a list of spatial weights (class 'listw')

other elements: NULL
```

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.

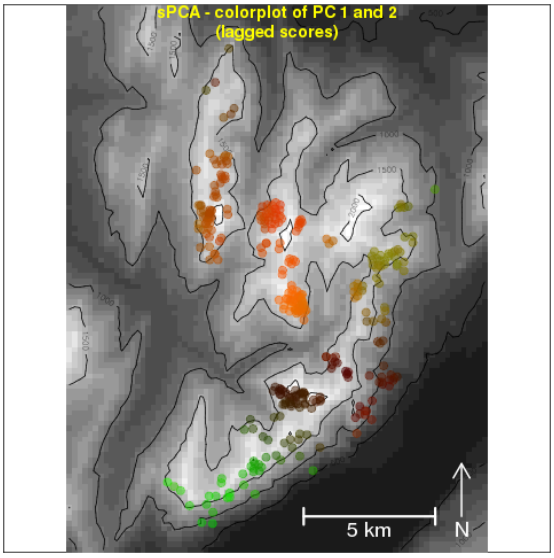
Try visualising the sPCA results as you did before with the PCA results. To clarify the possible spatial patterns, you can map the lagged PC ($ls) instead of the PC ($li), which are a 'denoisified' version of the PCs.

First, map the first principal component of sPCA. How would you interprete this result? How does it compare to the first PC of PCA? What inferrence can we make about the way the landscape influences gene flow in this population of Chamois?

Do the same with the second PC of sPCA. Some field observations suggest that this pattern is not artefactual. How would you interprete this second structure?

To finish, you can try representing both structures at the same time using the color coding introduced by [9] (`?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 = 2)
> 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. (2008) adegenet: a R package for the multivariate analysis of genetic markers. *Bioinformatics* 24: 1403-1405.

[2] 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.

[3] 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.

[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.

[9] Cavalli-Sforza LL, Menozzi P and Piazza A (1993) Demic expansions and human evolution. *Science* 259: 639–646.

[10] Callenge C (2006) The package "adehabitat" for the R software: a tool for the analysis of space and habitat use by animals. *Ecological Modelling* 197: 516–519.