


Tutorial using the  software

A tutorial for the spatial Principal Component Analysis using the R package *adegetnet_1.2-2*

T. JOMBART

This tutorial still is in its beta form: it may (and will surely) need completion and improvements. You can contribute to its improvement it by posting questions on the *adegetnet* forum: adegetnet-forum@lists.r-forge.r-project.org, or by sending comments to me at jombart@biomserv.univ-lyon1.fr.

Contents

1	Introduction	3
2	The sPCA in <i>adeget</i>	3
2.1	Some basics about sPCA	3
2.2	The <code>spca</code> function	4
2.3	A <code>spca</code> object	7
2.4	Representing the information	11
3	Diving into data: analysis of a real dataset	19

1 Introduction

This tutorial goes through the *spatial Principal Component Analysis* (sPCA, Jombart *et al.*, 2008), a multivariate method devoted to the multivariate analysis of genetic markers. The sPCA is implemented inside the *ade4* package (Jombart, 2008) for the R software (Ihaka & Gentleman, 1996; R Development Core Team, 2008). Parts of this implementation relies on functions from the *ade4* package (Chessel *et al.*, 2004; Dray *et al.*, 2007). Reading of the original paper describing the sPCA is assumed. The purpose of this tutorial is to provide guidelines for practical issues performing a sPCA and for interpreting the results. After recalling basics of the sPCA, we detail the different tools in *ade4* that are related to the method. We conclude by going through the analysis of an empirical dataset.

2 The sPCA in *ade4*

2.1 Some basics about sPCA

Mathematical notations used in this tutorial are those from Jombart *et al.* (2008). The sPCA analyses a data matrix \mathbf{X} which contains genotypes or populations (later referred 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, Legendre & Legendre, 1998, pp.572-576). 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 (Moran, 1948, 1950; Cliff & Ord, 1981):

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

In the case of genetic data, \mathbf{x} contains frequencies of an allele. Moran's I can be used to measure spatial structure among 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, and 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} :

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

$C(\mathbf{x})$ 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 $\psi = \mathbf{X}\mathbf{v}$) decomposing C from its maximum to its minimum value. Because $C(\mathbf{X}\mathbf{v})$ is a product of variance and 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 *adeget*.

2.2 The spca function

The simulated dataset used to illustrate this section has been analyzed in Jombart *et al.* (2008), and corresponds to the Figure 2A of the article. In *adeget*, the matrix of alleles frequencies previously denoted \mathbf{X} exactly corresponds to the `@tab` slot of `genind` or `genpop` objects:

```
> 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

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, 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, scannf = FALSE)
> mySpca2 <- spca(obj, xy = obj@other$xy, ask = FALSE, 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 **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 by Roger Bivand's package **spdep**. 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)
```

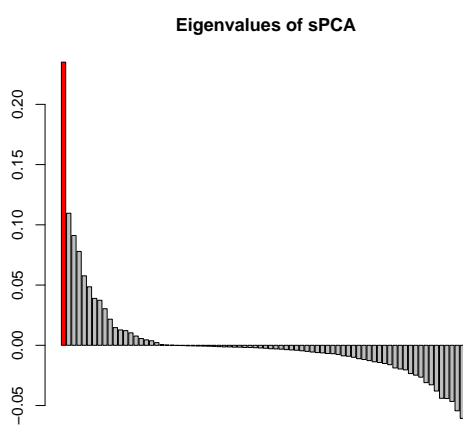
```
[1] "nb"
```

```
> 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`.

2.3 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
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 show how the elements described in Jombart *et al.* (2008) 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)
```

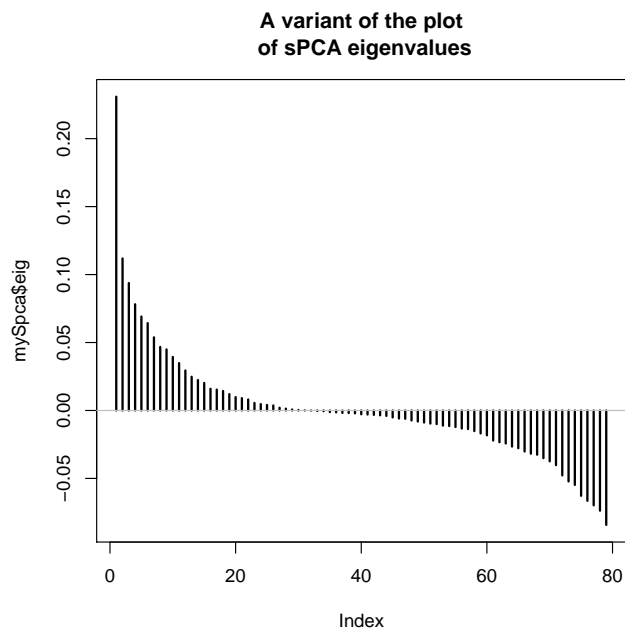
```
[1] -0.05480010 -0.06279067 -0.06647896 -0.06978457 -0.07375563 -0.08421213
```



```
> 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 \mathbf{v} in Jombart *et al.* (2008, eq.(4)) 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 -3.330669e-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 ($\psi = \mathbf{X}\mathbf{v}$), 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 displayed graphically instead of basic scores so as 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 \mathbf{u} in the paper) to the axes of the sPCA (\mathbf{v}). This is achieved by projecting \mathbf{u} onto \mathbf{v} , but this projection is a particular one: because both \mathbf{u} and \mathbf{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:

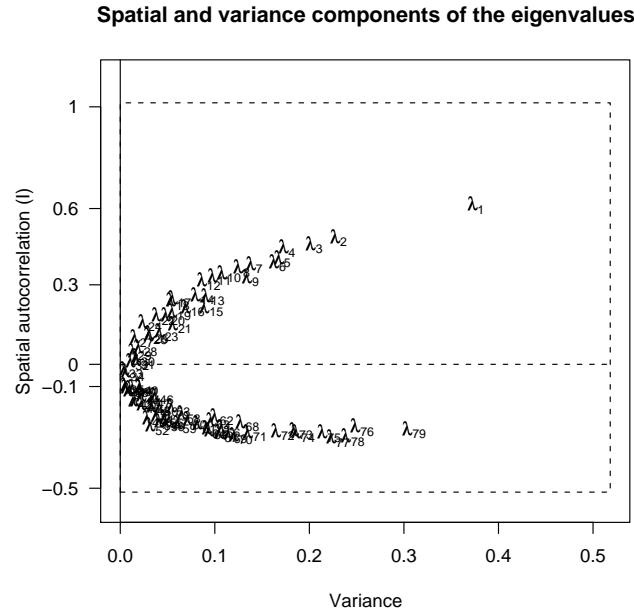
```
> mySpca$as
```

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

2.4 Representing the information

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 $i = 1, \dots, n-1$, where λ_1 is the strongest global eigenvalue, and λ_{n-1} is the strongest local eigenvalue) according to their variance and Moran's I components. These eigenvalues are contained inside a rectangle indicated by 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 Jombart *et al.* (2008) 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$stab, mySpca$lw, nperm = 99)
> myGtest
```

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

Observation: 0.01658103

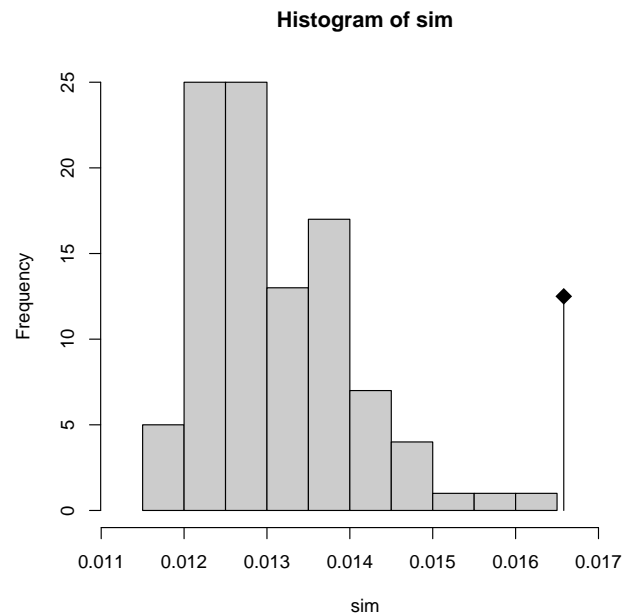
Based on 99 replicates

Simulated p-value: 0.01

Alternative hypothesis: greater

Std.Obs	Expectation	Variance
4.896271e+00	1.283647e-02	5.848858e-07

```
> plot(myGtest)
```



The produced object is a **randtest** object (see `?randtest`), which is the class of object 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 observed 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
```

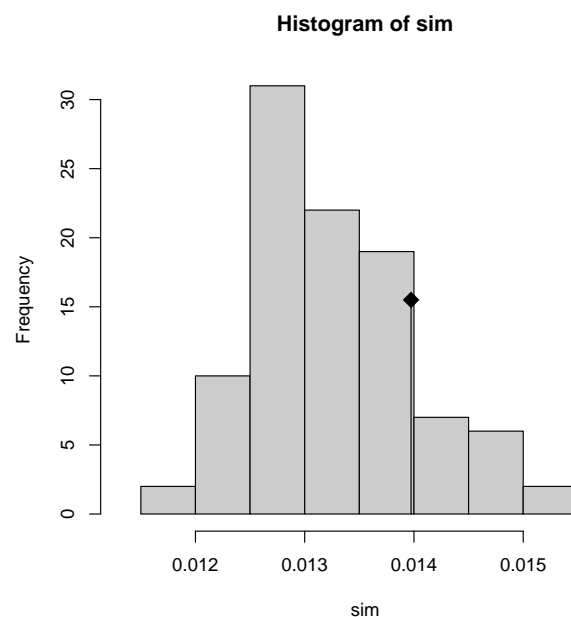
```
Monte-Carlo test
Call: local.rtest(X = obj$tab, listw = mySpca$lw, nperm = 99)

Observation: 0.01397349

Based on 99 replicates
Simulated p-value: 0.12
Alternative hypothesis: greater

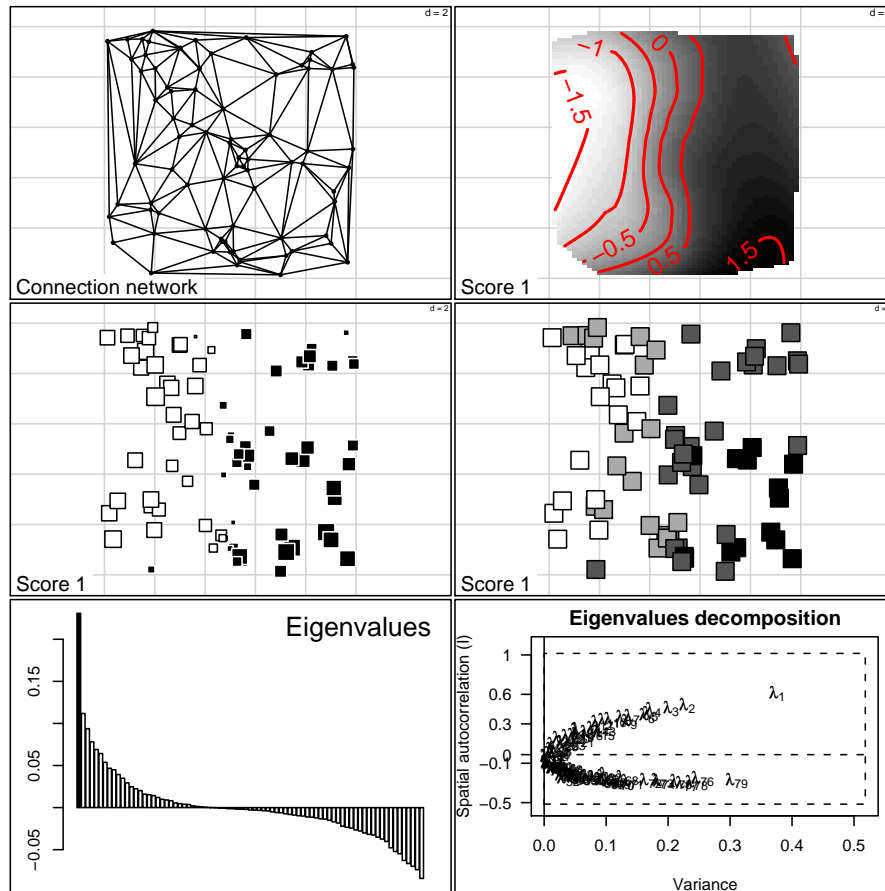
      Std.Obs  Expectation  Variance
1.134573e+00 1.317642e-02 4.935434e-07
```

```
> 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`):

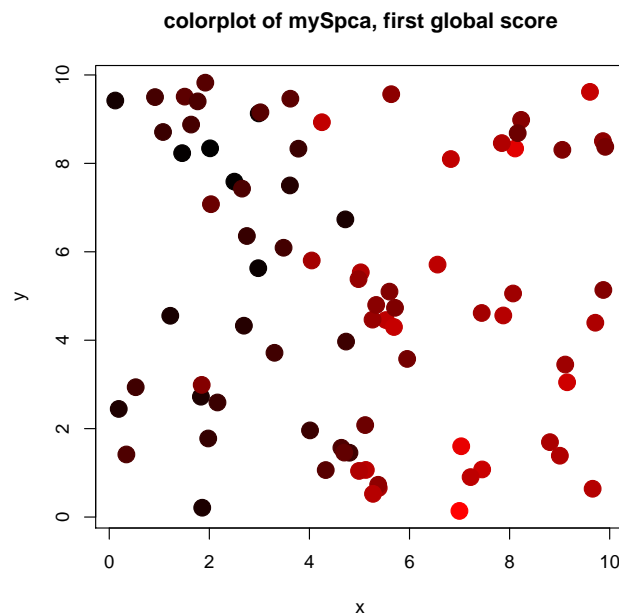
```
> plot(mySpca)
```



This figure shows different information, that we detail from the top to the bottom and from the left to the 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 entities scores onto one axis 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 steepest 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 Menozzi *et al.* (1978). Despite the `colorplot` clearly is more powerful to represent more than one set of scores on a single map, we can use it to represent the first global structure that was retained in `mySpca`:

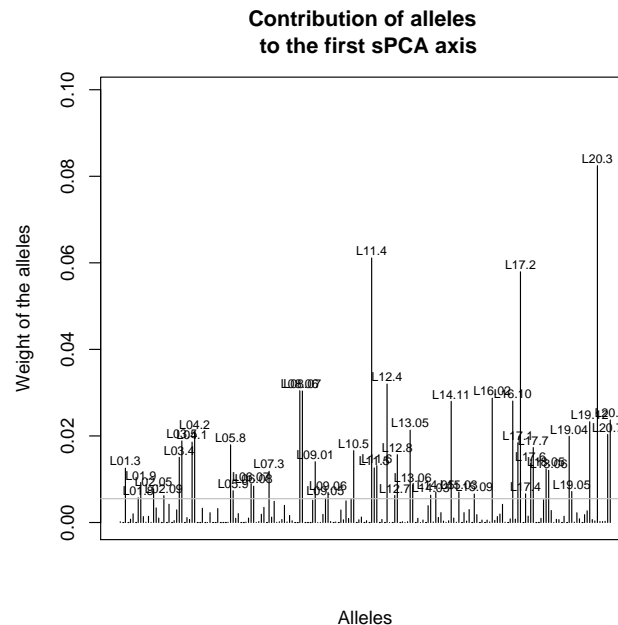
```
> colorplot(mySpca, cex = 3, main = "colorplot of mySpca, first global score")
```



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 set of scores. To quantify such contribution, the absolute value of loadings for a given structure can be used. 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 separate 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.lab = 1, cex.fac = 0.8)
> temp
```

```
$threshold
95%
0.02345973
```

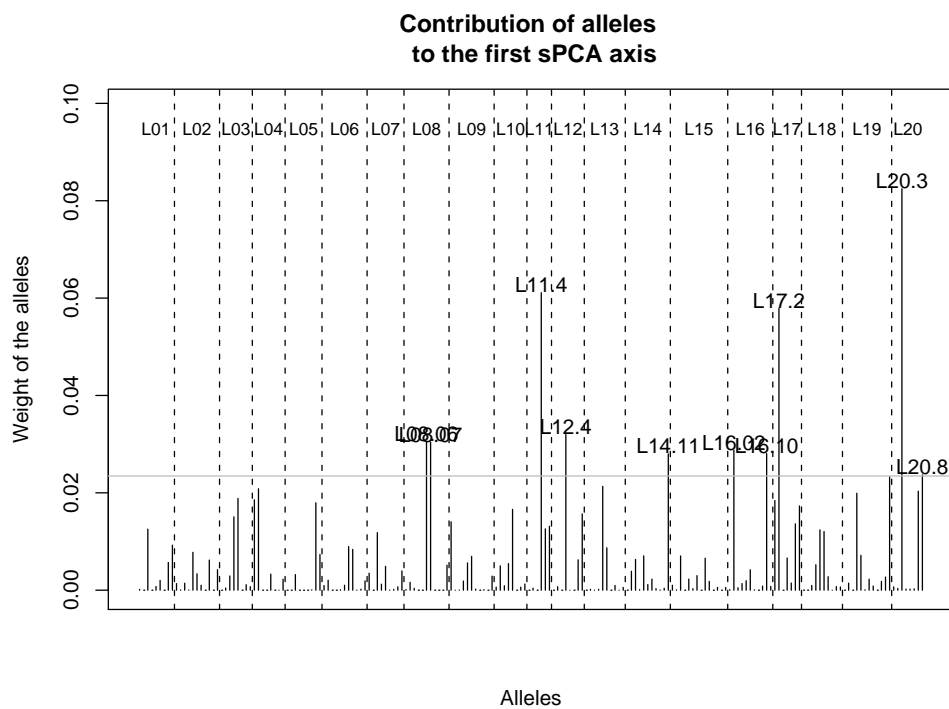
```
$var.names
[1] "L08.06" "L08.07" "L11.4" "L12.4" "L14.11" "L16.02" "L16.10" "L17.2"
[9] "L20.3" "L20.8"
```

```

$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

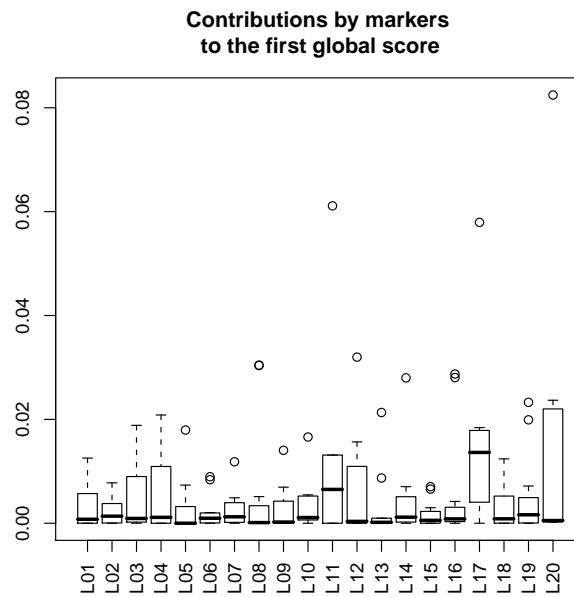
$var.values
      L08.06      L08.07      L11.4      L12.4      L14.11      L16.02      L16.10
0.03044687 0.03037709 0.06111338 0.03199067 0.02799529 0.02873923 0.02806079
      L17.2      L20.3      L20.8
0.05793290 0.08246000 0.02368209

```



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 glob
```



3 Diving into data: analysis of a real dataset

To come...

References

- CHESSEL, D., DUFOUR, A.-B. & THIOULOUSE, J. (2004). The ade4 package-I: one-table methods. *R News* **4**, 5–10.
- CLIFF, A. & ORD, J. (1981). *Spatial Processes. Model & Applications*. London: Pion.
- DRAY, S., DUFOUR, A.-B. & CHESSEL, D. (2007). The ade4 package - II: Two-table and K -table methods. *R News* **7**, 47–54.
- IHAKA, R. & GENTLEMAN, R. (1996). R: A language for data analysis and graphics. *Journal of Computational & Graphical Statistics* **5**, 299–314.
- JOMBART, T. (2008). adegenet: a R package for the multivariate analysis of genetic markers. *Bioinformatics* **24**, 1403–1405.
- JOMBART, T., DEVILLARD, S., DUFOUR, A.-B. & PONTIER, D. (2008). Revealing cryptic spatial patterns in genetic variability by a new multivariate method. *Heredity* **101**, 92–103.
- LEGENDRE, P. & LEGENDRE, L. (1998). *Numerical ecology*. Elsevier Science B.V., Amsterdam.
- MENOZZI, P., PIAZZA, A. & CAVALLI-SFORZA, L. (1978). Synthetic maps of human gene frequencies in Europeans. *Science* **201**, 786–792.
- MORAN, P. (1948). The interpretation of statistical maps. *Journal of the Royal Statistical Society, B* **10**, 243–251.
- MORAN, P. (1950). Notes on continuous stochastic phenomena. *Biometrika* **37**, 17–23.
- R DEVELOPMENT CORE TEAM (2008). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL <http://www.R-project.org>. ISBN 3-900051-07-0.