Ecography

E5567

Jombart, T., Dray, S. and Dufour, A.-B. 2009. Finding essential scales of spatial variation in ecological data: a multivariate approach. – Ecography 32: 161–168.

Supplementary material

Appendix 1. Computations of the multi-scale pattern analysis (MSPA)

This appendix details the computations of the MSPA as summarized by Figure 1 in the article.

Data transformations

The raw data matrix **X** is transformed as follows: qualitative variables are replaced by sets of dummy vectors, and then all vectors (i.e., quantitative variables and dummy vectors) are centred and scaled. The obtained matrix is denoted **Y**.
The vector $\mathbf{d}=[d_i]$ $(i = 1, q)$ containing weights of the columns of **Y** is computed as follows:
– $d_i=1/q$ if the column $\mathbf{y}_i$ is a quantitative variable.
– $d_i=k_i/(qn)$ if the column $\mathbf{y}_i$ is the dummy vector of a modality observed in k_i sites.

If a canonical MSPA is performed, **Y** is regressed onto a n × r matrix of covariates, denoted **C**.
The predicted matrix $\hat{\mathbf{Y}}$ is computed as:
$\hat{\mathbf{Y}} = \mathbf{C}(\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T\mathbf{Y}$
In canonical MSPA, $\hat{\mathbf{Y}}$ replaces **Y** in the remaining of the computations.
In partial canonical MSPA, $(\mathbf{Y} – \hat{\mathbf{Y}})$ replaces **Y** in the remaining of the computations.

The matrix **U** is obtained by the eigen analysis of the adjacency matrix of a connection network. Its columns $\mathbf{u}_j$ $(j = 1, n–1)$ are the centred and scaled Moran's eigenvector maps (MEMs).

Regression of the variables onto MEMs

Each variable $\mathbf{y}_i$ is regressed on each MEM $\mathbf{u}_j$, yielding a matrix **S** containing q × n–1 determination coefficients (R^2) between variables and MEMs. **S** is computed as:
$\mathbf{S} = ((\mathbf{Y}^T\mathbf{U})*(\mathbf{Y}^T\mathbf{U})) / (n^2)$
where $\mathbf{Y}^T$ is the transposed matrix of **Y** and where "*" denotes the Hadamard product.

Centering of S

The centering terms, denoted $E(R^2(\mathbf{y}_i,\mathbf{u}_j))$, are the values of R^2 expected when variables in **Y** are not structured at any scale. These terms equal 1/(n–1) if column vectors of **Y** are normally distributed, yielding the new matrix **Z** computed as:
$\mathbf{Z} = \mathbf{S} – (\mathbf{1}_q\mathbf{1}_{n-1}^T)/(n–1)$
where $\mathbf{1}_q$ is the q-dimensional vector whose components are all one.
When column vectors of **Y** are not normally-distributed, $E(R^2(\mathbf{y}_i,\mathbf{u}_j))$ can be obtained in a non-parametrical way.
Rows of **Y** are randomly permuted and R^2 are computed, yielding one value of $E(R^2)$ for each term in **S**.
This operation is repeated a large number of times (1000 by default), yielding a distribution of $E(R^2)$ for each term in **S**.
The mean of these distributions are taken to obtain the final centering values, that are subtracted to **S** to obtain a matrix **Z**.
Because only positive deviations from $E(R^2)$ are of interest, negative deviations are remapped to zero.

% principal component analysis of Z

The MSPA is the %PCA of **Z** using the components of **d** as row weights.
Let **D** be the diagonal matrix of row weights, defined as $\mathbf{D} = \text{diag}(d_1, ..., d_i , ..., d_q)$.
The principal axes of MSPA (denoted **A**) are given by the eigen analysis of $\mathbf{Z}^T\mathbf{D}\mathbf{Z}$, that is:

$\mathbf{Z}^T\mathbf{DZA} = \mathbf{AL}$

where **L** is a diagonal matrix of eigenvalues.

Differently from the usual PCA, the principal components of %PCA (denoted **B**) are obtained by projecting the non-centred data onto the principal axes, that is:

$\mathbf{B} = \mathbf{SA}$

The principal components associated to the highest eigenvalues represent the largest parts of inertia, that is, of squared distances among the scale profiles of all variables. These principal components therefore separate as well as possible the variables according to their multiscale spatial patterns.

Appendix 2. Code in R language of the MSPA function (`mspa`) and the associated graphical function (`scatter.mspa`)

These functions are designed to be used with R 2.7.1 <http://www.r-project.org/>, and require the packages `ade4` and `spdep`.

The arguments are intended as follows:
– `dudi`: a duality diagram (i.e. a reduced space ordination) obtained by a dudi function (for instance dudi.pca)
– `lw`: a list of spatial weights of class listw
– `scannf`: logical, indicating whether the screeplot should be displayed to choose the number or retained factors
– `nf`: the number of retained factors
– `centring`: a character string indicating if parametric `("param")` or non-parametric `("sim")` centring should be used
– `nperm`: an integer giving the number of permutations used to compute the theoretical coefficients of determination (1000 by default); used if `centring="sim"`
– `x`: a mspa object
– `xax`: an integer indicating the x axis to be displayed
– `yax`: an integer indicating the y axis to be displayed
– `clab.var`: a number indicating the size of the labels of variables
– `clab.sca`: a number indicating the size of the labels of scales
– `posieig`: a character indicating the position of the screeplot (any of the four combination between "top", "bottom", "left" and "right")
– `sub`: an optional character added as legend to the plot
– `ratio`: a number between 0 and 1 indicating the size of the screeplot as a proportion of the plot
– `bary`: a logical indicating whether the barycenter of the variables should be displayed
– `circle`: a logical indicating whether a circle of radius one should be displayed
– `...`: further arguments to be passed to other methods.

```
############
## R code ##
############

#####################
# Function mspa
#####################
mspa <- function(dudi, lw, scannf = TRUE, nf = 2,
centring=c("param","sim"), nperm=1000){

  # arguments checks
  if(!require(ade4)) stop("package ade4 is required")
  if(!require(spdep)) stop("package spdep is required")
  if (!inherits(dudi, "dudi")) stop("object of class 'dudi' expected")
  if(!inherits(lw,"listw")) stop("lw must be a listw object (package
spdep)")
  if(lw$style != "W") stop("lw must be row-nowmalized (style W)")

  df <- dudi$tab
  varweights <- dudi$cw/sum(dudi$cw)
  n <- nrow(df)
  p <- ncol(df)
  centring <- match.arg(centring)

  ## auxiliary variables and functions

  # retrieve var.idx with correct variable names
  findname <- function(vec){
```

```
    if(length(vec) == 1) return(vec)
    res <- sub("[.][^.]*$","",vec[1])
    return(res)
  }

  # var.idx
  var.idx <- dudi$assign
  if(!is.null(var.idx)){
    temp <- split(colnames(df), var.idx)
    newlev <- sapply(temp, findname)
    levels(var.idx) <- newlev
  }

  genlab <- function(base,n){
    f1 <- function(cha, n) {
      if (nchar(cha) < n) {
        cha <- paste("0", cha, sep = "")
        return(f1(cha, n))
      }
      return(cha)
    }
    w <- as.character(1:n)
    max0 <- max(nchar(w))
    w <- sapply(w, function(cha) f1(cha, max0))
    return(paste(base, w, sep = ""))
  } # end genlab


  # matrix centring and scaling
  X <- scalewt(df, center=TRUE, scale=TRUE)

  # computation of centred and scaled eigenvectors of the connection
network
  # denoted U
  U <- as.matrix(orthobasis.listw(lw))
  r <- ncol(U)
  colnames(U) <- genlab("u_",r)

  ## projection of X onto U
  ## only R-squared of each vector of U are kept

  ## model computations
  R <- t(X) %*% U /n
  R2 <- R*R

  ## handle centring of R2
  if(centring=="param"){
      newdf <- as.data.frame(R2-(1/(n-1)))
  } else{ # i.e. if centring is non-parametric
      tempX <- t(X)
      fPerm <- function(X){
          permX <- X[, sample(1:n)] # X has to be transposed, that is,
variables in rows, obs in columns
          res <- permX %*% U /n
          res <- res * res
          return(res)
      } # end fPerm
      listR2sim <- lapply(1:nperm, function(i) fPerm(tempX))
      meanR2sim <- listR2sim[[1]]
      for(i in 2:nperm){
          meanR2sim <- meanR2sim + listR2sim[[i]]
```

```
      } # end for

      meanR2sim <- meanR2sim / nperm

      newdf <- as.data.frame(R2 - meanR2sim)
  } # end non-parametric centring

  ## keep only positive deviation in R2
  newdf[newdf < 0] <- 0

  ## we proceed to the analysis of this matrix
  res <- as.dudi(newdf, scannf=scannf, nf=nf, row.w = varweights,
                 col.w = rep(1, r), call=match.call(), type="mspa")

  res$ls <- as.data.frame(as.matrix(R2) %*% as.matrix(res$c1))
  colnames(res$ls) <- colnames(res$li)
  row.names(res$ls) <- row.names(res$li)

  xmoy <- apply(R2, 2, function(c) weighted.mean(c,varweights))
  bary <- as.vector(t(xmoy) %*% as.matrix(res$c1))
  names(bary) <- colnames(res$c1)

  res$R2 <- R2
  res$meanPoint <- bary
  res$varweights <- varweights
  names(res$varweights) <- colnames(X)
  if(!is.null(var.idx)) res$assign <- var.idx
  if(centring=="sim") {
      res$centring <- meanR2sim
  }

  # return result
  return(res)
}



##############################
# function scatter.mspa
##############################
scatter.mspa <- function(x, xax = 1, yax = 2, clab.var = 0.75, clab.sca =
1,
                         posieig = "top", sub = NULL, ratio = 1/4,
bary=TRUE, circle=TRUE, ...){

  if(!inherits(x,"mspa")) stop("to be used with mspa objects only")

  opar <- par(mar = par("mar"))
  on.exit(par(opar))

  s.arrow(x$c1[,c(xax,yax)], clab=clab.sca, sub=sub, ...)

  extrem <- chull(x$c1[,c(xax,yax)])

  par(xpd=TRUE)
  polygon(x$c1[extrem,c(xax,yax)],col="lightgrey")
  if(circle) {symbols(x=0,y=0,circles=1,add=TRUE,inches=FALSE)}

  s.arrow(x$c1[,c(xax,yax)], clab=clab.sca, add.p=TRUE)
  s.label(x$ls[,c(xax,yax)], clab=clab.var, add.p=TRUE)
```

```
  # compute coordinates of factors
  # = weighted mean of its levels
  if(!is.null(x$assign)) {
    temp <- apply(x$li[,c(xax,yax)],2,function(c) tapply(c, x$assign,mean))
    rownames(temp) <- unique(x$assign)
    s.label(temp, add.plot=TRUE, clab=clab.var)
  }

  if(bary) points(x$meanPoint[xax],x$meanPoint[yax],pch=20,cex=2)

  add.scatter.eig(x$eig, x$nf, xax, yax, posi = posieig, ratio = ratio)

  par(mar=rep(.1,4))
  box(which="figure")

  return(invisible(match.call))
}
```

Appendix 3. Code in R language of the presented illustrations

The R code presented in Appendix 2 is assumed to be saved in the file "mspa.R". The R packages `ade4`, `spdep` and `adegenet` are required. Note that as simulated data involve a random process, results may slightly differ from one simulation to another.

```
# source mspa functions
source("mspa.R")

# load packages
library(ade4)
library(spdep)
library(adegenet)



#################
# simulated data
#################

# spatial coordinates
xy <- expand.grid(1:10,1:10)
lw <- nb2listw(cell2nb(10,10))

# MEMs
U <- orthobasis.listw(lw)

# structured variables
V <- list()
# large scale
V$V1 <- 0.5*U[,1] + 0.5*U[,2] + 0.5*U[,3] + rnorm(100)
V$V2 <- 0.5*U[,1] - 0.8*U[,2] + 0.5*U[,3] + rnorm(100)
V$V3 <- U[,1] - U[,2] + 0.5*U[,3] + rnorm(100)

# medium scale
V$V4 <- 0.6*U[,44] + U[,45] +  0.8*U[,46] + rnorm(100)

# fine scale
V$V5 <-  0.5*U[,97] + U[,98] + U[,99]  + rnorm(100)
V$V6 <- 0.5*U[,97] + 0.5*U[,98] - U[,99] + rnorm(100)
V$V7 <- 0.6*U[,97] + 0.6*U[,98] + 0.8*U[,99]  + rnorm(100)

# complete dataset (V1 -> V7 and 28 random variables
V <- as.data.frame(V)
M <- matrix(rnorm(100*28),nrow=100)
X <- cbind.data.frame(V,M)
colnames(X) <- paste("V",1:ncol(X),sep='')

# the mspa function requires a dudi object
# here we use a PCA
pcaX <- dudi.pca(X,scannf=FALSE,nf=4)

# mspa
mspa1 <- mspa(pcaX,lw,scannf=F,nf=3)

# figure
scatter(mspa1)
scatter(mspa1,2,3)
```

```
################
# oribatid data
################

# function to proceed to the Hellinger transformation
hellTrans <- function(X){
  if (!( is.matrix(X) | is.data.frame(X) )) stop("Object is not a matrix or
data frame.")
  if (any(is.na(X))) stop("na entries in table.")

  sumRow <- apply(X,1,sum)
  Y <- X/sumRow
  Y <- sqrt(Y)

  return(Y)
}

# load oribatid data
data(oribatid)

# choose a connection network
cn <- chooseCN(oribatid$xy,res="listw",ask=FALSE,type=1)$cn

# analysis of environmental variables
# using the analysis of Hill & Smith
# for mixed data (quantitative, qualitative)
hsEnv <- dudi.hillsmith(oribatid$envir,scannf=FALSE)

# detrending of the data (residuals from projection on xy coordinates)
hsEnv.detr <- pcaivortho(hsEnv,oribatid$xy,scannf=FALSE)

# MSPA of environmental data
mspaEnv <- mspa(hsEnv.detr,cn,scannf=FALSE,nf=2)
scatter(mspaEnv)

# analysis of species data (residuals from projection on xy coordinates)
pcaSpe <- dudi.pca(hellTrans(oribatid$fau),scale=FALSE,scannf=FALSE)

# detrending of the data
pcaSpe.detr <- pcaivortho(pcaSpe,oribatid$xy,scannf=FALSE)

# MSPA of species data
mspaSpe <- mspa(pcaSpe.detr,cn,scannf=FALSE,nf=2)
scatter(mspaSpe)

# RDA of species predicted by environment
rda1 <- pcaiv(dudi=pcaSpe.detr, df=oribatid$envir,scannf=FALSE,nf=2)

# mspa of species predicted by environment
mspaSpePred <- mspa(dudi=rda1, lw=cn, scannf=FALSE, nf=2)
scatter(mspaSpePred)

# compute projections of environmental variables onto principal axes
supEnv <- as.matrix(mspaEnv$R2) %*% as.matrix(mspaSpePred$c1)
s.arrow(mspaSpePred$c1)
s.label(supEnv,1,2,add.p=TRUE)
```

Appendix 4. Description of simulated data

The simulated dataset was made of 35 variables (V1–V35) observed at 100 sites on a 10 by 10 regular grid, connected following the rook connections (neighbours share an edge).
Seven variables (V1–V7) were spatially structured, the other variables being drawn randomly from a normal variate distribution.
The structured variables were obtained by linear combinations of MEMs (U_1–U_{99}) with addition of random noise (normal variate distribution, mean=0, SD=1).
The coefficients of these combinations are given in the following tables.
Structured variables are graphically represented in Fig. S4:1.

Table S4:1. Large-scale structured variables

	U_1	U_2	U_3
V1	0.5	0.5	0.5
V2	0.5	–0.8	0.5
V3	1	–1	0.5

Table S4:2. Medium-scale structured variable

	U_{44}	U_{45}	U_{46}
V4	0.6	1	0.8

Table S4:3. Fine-scale structured variables

	U_{97}	U_{98}	U_{99}
V5	0.5	1	1
V6	0.5	0.5	–1
V7	0.6	0.6	0.8

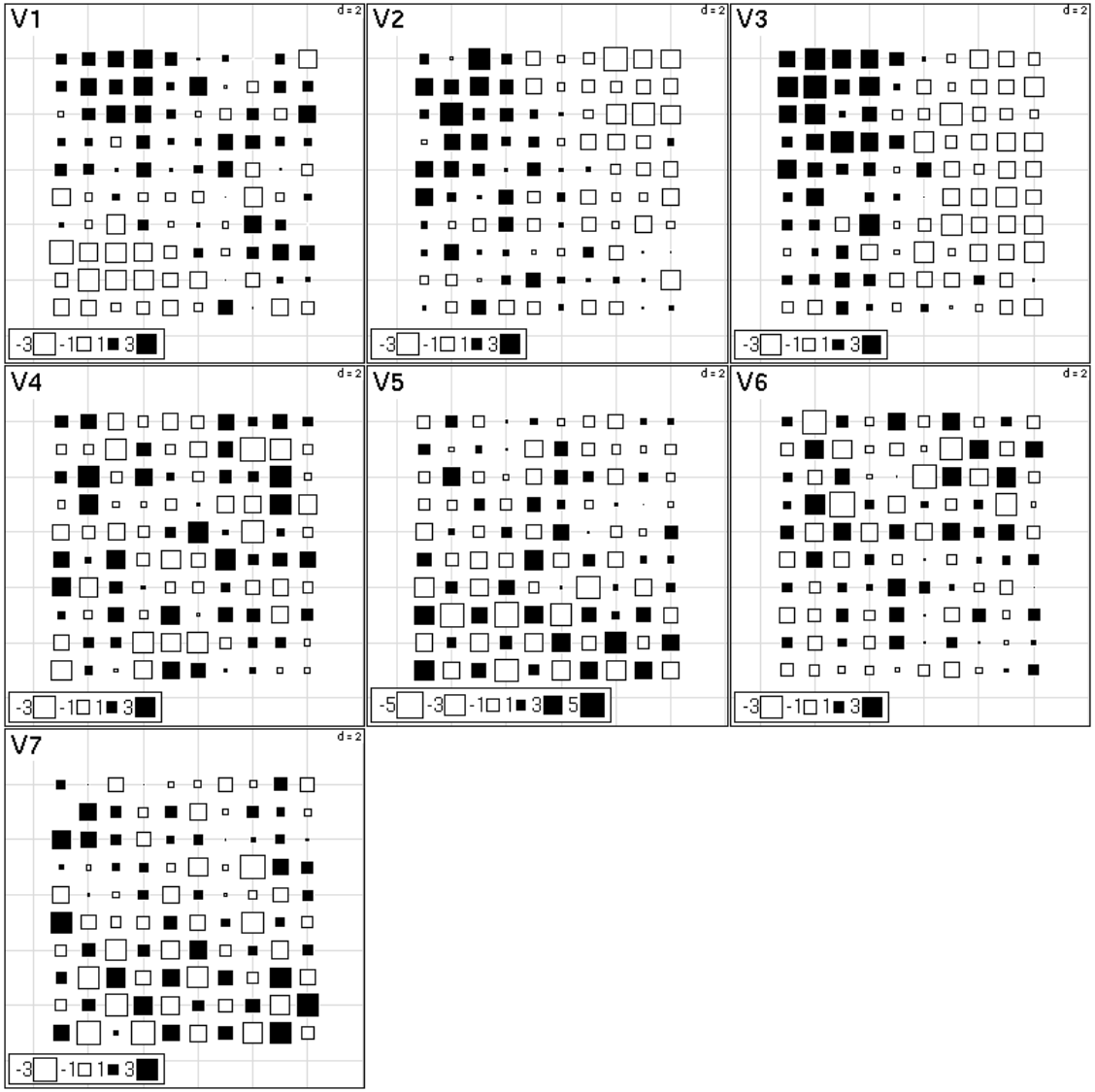

Figure S4:1. Graphical representation of structured variables (V1–V7).

Appendix 5. Forward selection of simulated data

This table gives the results of the forward selection of the multivariate regression of simulated data (Appendix 4) on MEMs vectors. All the presented vectors were retained. The forward selection was performed using the R package packfor proposed by Stéphane Dray <http://biomserv.univ-lyon1.fr/~dray/software.php>.

variables	R^2	R^2 (Cumulated)	Adjusted R^2 (Cumulated)	F	pval
VP99	0.07	0.07	0.06	7.88	0.001
VP2	0.04	0.12	0.10	4.78	0.001
VP45	0.03	0.15	0.13	3.88	0.001
VP98	0.03	0.19	0.15	3.99	0.001
VP1	0.03	0.22	0.18	3.77	0.001
VP97	0.03	0.24	0.20	3.35	0.001
VP3	0.03	0.27	0.22	3.27	0.001
VP46	0.02	0.29	0.23	2.68	0.001
VP44	0.02	0.31	0.24	2.04	0.001
VP88	0.01	0.32	0.25	1.91	0.002
VP42	0.01	0.33	0.25	1.64	0.011
VP48	0.01	0.35	0.26	1.59	0.016
VP20	0.01	0.36	0.26	1.56	0.018
VP16	0.01	0.37	0.27	1.53	0.031