N° d'ordre : 293-2004 Année : 2004-2005

THESE

présentée

devant l'UNIVERSITE CLAUDE BERNARD - LYON 1

pour l'obtention du DIPLOME DE DOCTORAT
(arrêté du 25 avril 2002)

et soutenue publiquement le

20 décembre 2004

par

OLLIER Sébastien

Des outils pour l'intégration des contraintes spatiales, temporelles et évolutives en analyse des données écologiques

Tome 2 : annexes

Spécialité : biostatistique

JURY : Dominique Pontier, Présidente
Nigel Yoccoz, Rapporteur
Claude Millier, Rapporteur
Jean Thioulouse, Directeur
Pierre Couteron, Co-directeur

SOMMAIRE

LES DONNÉES

1. Répartition spatiale de l'espèce *Acacia ehrenbergiana*

alias

acacia

description

'acacia' donne l'abondance de l'espèce *Acacia ehrenbergiana* pour cinq transects formés de 32 cadrats.

format

'acacia' est data frame à 15 colonnes et 32 lignes. Chaque colonne représente l'abondance de l'espèce dans un des 5 transects pour l'une des trois catégories échantillonnées (**se**edlings, **sm**all trees (crown < 1m^2 in canopy), **la**rge trees (crown > 1m^2 in canopy)).

sources

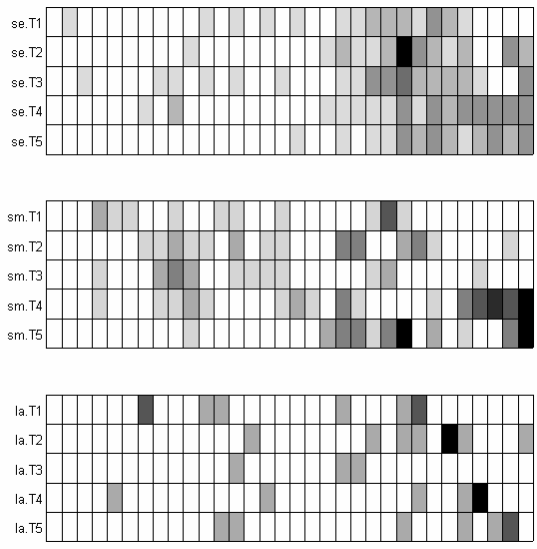
Greig-Smith, P. & Chadwick, M.J. (1965) Data on pattern within plant communities. III. *Acacia-Capparis* semi-desert scrub in the Sudan. Journal of Ecology, 53, 465-474.

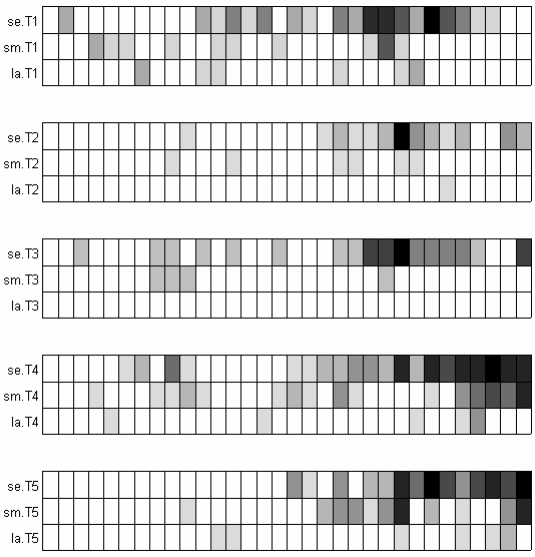
références

Hill, M.O. (1973) The intensity of spatial pattern in plant communities. Journal of Ecology, 61, 225-235.

exemples

```
# représentation par catégories
par(mfrow = c(3, 1))
fac <- as.factor(rep(1:3,
 rep(5,3)))
u <- split(as.data.frame(
 t(sqrt(acacia))), fac)
lapply(u, function(x) table.paint(
 x, cleg = 0, clabel.row = 2,
 clabel.col = 0))


# représentation par transect
par(mfrow = c(5, 1))
fac <- as.factor(rep(1:5, 3))
u <- split(as.data.frame(
 t(sqrt(acacia))), fac)
lapply(u, function(x) table.paint(
 x, cleg = 0, clabel.row = 2,
 clabel.col = 0))
```





2. Variabilité génétique des populations de Cacadors

alias

atya

description

'atya' est un jeu de données portant sur la variabilité génétique d'une population de Cacador (*atya innocous*) de l'île de Basse-Terre en Guadeloupe. Cette crevette vit et se reproduit en eau douce mais les larves dévalent et ont une période de croissance en mer. Les données rassemblent les coordonnées spatiales de 31 stations pour lesquelles des données génétiques (6 locus) sont disponibles. Un fond de carte est également disponible afin de représenter les données.

format

'atya' est une liste à 3 composantes:

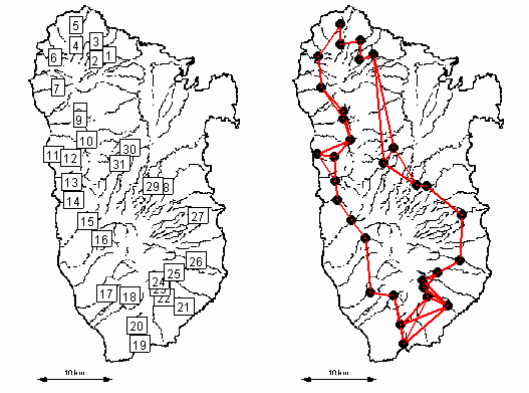
- *xy* : est un data frame dont les 31 lignes correspondent aux coordonnées spatiales des stations.
- *gen* : est un data frame à 31 lignes et 22 colonnes. Chaque case contient la fréquence allélique pour une station et un locus considérés. On a 6 loci avec respectivement 2, 5, 2, 4, 4, et 5 allèles.
- *neig* : est un graphe de voisinage. Deux stations sont considérées comme proches si elles sont dans le même bassin versant ou si elles sont dans deux bassins versants voisins au sens de la distance à parcourir le long de la côte entre les embouchures.

sources

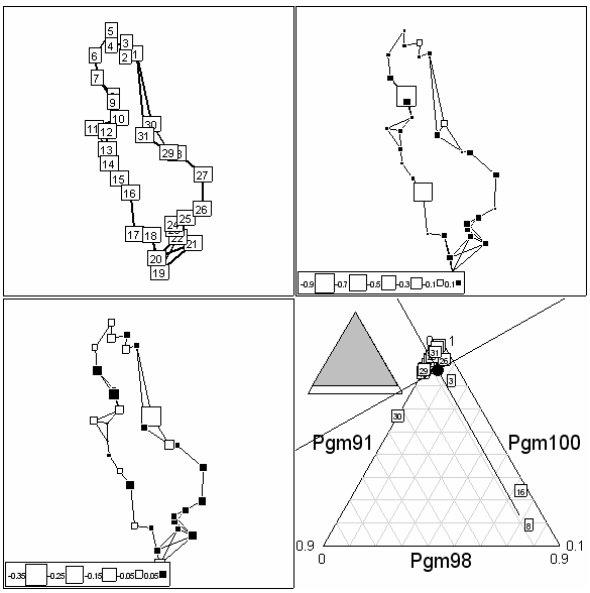
Fievet, E., Eppe, F., & Dolédec, S. (2001). Etude de la variabilité morphométrique et génétique des populations de Cacadors (*Atya innocous* et *Atya scabra*) de l'île de Basse-Terre. Direction Régionale de L'Environnement Guadeloupe, Laboratoire des hydrosystèmes fluviaux, Université Lyon 1, 43 Bd du 11 Novembre 1918, 69622, Villeurbanne cedex, France.

exemples

```
at.xy <- atya$xy
at.gen <- atya$gen
u <- as.character(substitute(donnees))
dirlib <- getwd()
file <- file.path(dirlib, u, "atyacarto.pnm")
at.pnm <- read.pnm(file)
at.nb <- neig2nb(atya$neig)
at.listw <- nb2listw(at.nb)
plot(at.pnm)
s.label(at.xy, add.plot = T, clab = 0.75)
plot(at.pnm)
points(at.xy, pch=20,cex=2)
plot(at.nb, at.xy, col = "red",
 add = T, lwd = 2)
```



```
at.pca <- dudi.pca(at.gen, scale = F)
Select the number of axes: 2
s.arrow(at.pca$c1, clab = 2)
par(mfrow = c(2, 2))
s.label(at.xy, neig = atya$neig,
 grid = FALSE, addaxes = FALSE, inc = F)
s.value(at.xy, at.pca$li[,1],
 neig = atya$neig, grid = FALSE,
 addaxes = FALSE, inc = F)
s.value(at.xy, at.pca$li[,2],
 neig = atya$neig, grid = FALSE,
 addaxes = FALSE, inc = F)
triangle.plot(at.gen[,19:21],
 clab = 0.75, addaxes = T)
```



3. Traits biologiques et écologiques d'insectes aquatiques

alias

bsetal97

description

'bsetal97' fournit les traits biologiques et écologiques de 131 espèces d'insectes aquatiques, ainsi que la taxonomie des espèces.

format

'bsetal97' est une liste à 8 composantes :

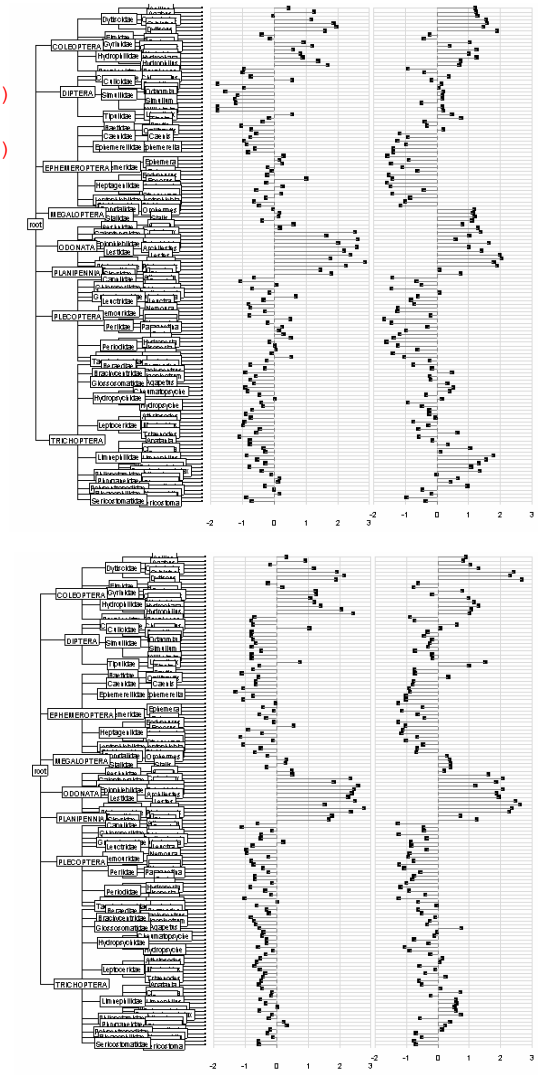
- *species.names* : est un vecteur de chaînes de caractères dont chaque élément correspond au nom des 131 espèces
- *taxo* : est un onjet de la classe 'taxo'
- *biol* : est un data frame contenant les 10 traits biologiques
- *biol.blo* : est un vecteur dont chaque élément correspond au nombre d'insectes sur lesquels chaque trait a été mesuré
- *biol*.blo.names : est un vecteur de chaînes de caractères dont chaque élément correspond au nom des 10 traits biologiques
- *ecol* : est un data frame contenant les 7 traits écologiques
- *ecol.blo* : est un vecteur dont chaque élément correspond au nombre d'insectes sur lesquels chaque trait a été mesuré
- *ecol*.blo.names : est un vecteur de chaînes de caractères dont chaque élément correspond au nom des 7 traits écologiques

sources

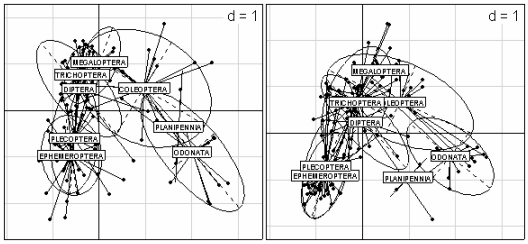
Statzner, B., Hoppenhaus, K., Arens, M.-F., & Richoux, P. (1997) Reproductive traits, habitat use and templet theory: a synthesis of world-wide data on aquatic insects. Freshwater Biology, 38, 109-135.

exemples

```
# la taxonomie
bsetal.phy <- taxo2phylog(as.taxo(bsetal97$taxo))
# les traits biologiques
X <- prep.fuzzy.var(bsetal97$biol,bsetal97$biol.blo)
# les traits écologiques
Y <- prep.fuzzy.var(bsetal97$ecol,bsetal97$ecol.blo)
# analyse des correspondances floues de X
dudi1 <- dudi.fca(X, scan = F)
# analyse des correspondances floues de Y
dudi2 = dudi.fca(Y,scan=F)
# score1 de l'analyse de X
a1 <- dudi1$l1[,1]
names(a1) <- bsetal97$species.names
# score1 de l'analyse de Y
a2 <- dudi2$l1[,1]
names(a2) <- bsetal97$species.names
values <- cbind.data.frame(a1, a2)
# chacun des deux premiers scores est clairement
# dépendant de la taxonomie
dotchart.phylog(bsetal.phy, values, ceti = 0.5,
 csub = 0, cdot = 0.65, cleaves = 0.3,
 clabel.nodes = 0.5)

# analyse de coinertie : étude de l'association
# entre les deux tableaux
coin <- coinertia(dudi1, dudi2, scan = FALSE)
# score1 normalisé des lignes de X
a1 <- coin$mX[,1]
names(a1) <- bsetal97$species.names
# score1 normalisé des lignes de Y
a2 <- coin$mY[,1]
names(a2) <- bsetal97$species.names
values <- cbind.data.frame(a1, a2)
dotchart.phylog(bsetal.phy, values, ceti = 0.5,
csub = 0, cdot = 0.65, cleaves = 0.3,
 clabel.nodes = 0.5)
```



```
par(mfrow = c(1,2))
s.class(coin$mX,
 bsetal97$taxo$ord,
 clab = 0.5)
s.class(coin$mY,
 bsetal97$taxo$ord,
 clab = 0.5)
# l'association entre les deux tableaux
# se fait directement au niveau de l'ordre
```



4. Taxonomie, phylogénie et traits biologiques de 49 mammifères

alias

carniherbi49

description

'carniherbi49' fournit les traits biologiques, la phylogénie et la taxonomie de 49 mammifères.

format

'carniherbi49' est une liste à 5 composantes :

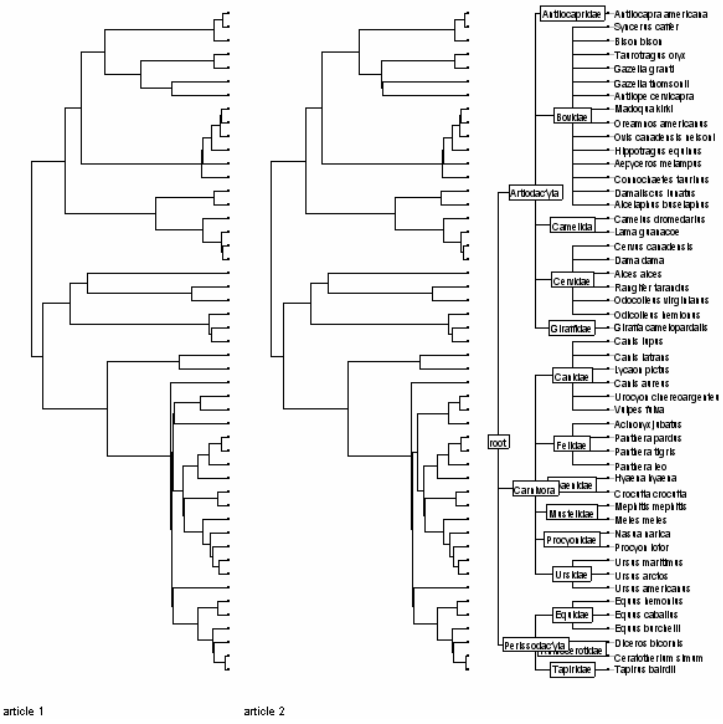
- *taxo* : est un data.frame définissant la taxonomie des espèces (fam : facteur à 14 classes correspondant aux familles, ord : facteur à 3 classes correspondant aux ordres)
- *tre1* : est un arbre phylogénétique au format 'Newick' définit dans (Garland et al., 1993)
- *tre2* : est un arbre phylogénétique au format 'Newick' définit dans (Garland & Janis, 1993) (la longueur de la racine aux feuilles vaut à 70 millions d'années)
- *tab1* : est un data.frame de traits biologiques à 49 lignes et 2 colonnes :
  1. bodymass : correspond au poids du corps (kg)
  2. homerange : correspond à l'espace vital (km^2)
- *tab2* : est un data.frame de traits biologiques à 49 lignes et 5 colonnes :
  1. clade : donne le régime alimentaire (30 carnivores et 19 herbivores)
  2. runningspeed : donne la vitesse maximale de course en km/h
  3. bodymass : donne le poids du corps (kg)
  4. hindlength : donne la taille des membres postérieurs (cm)
  5. mtfratio : donne le rapport métatarse/fémur

sources

Garland, T., Dickerman, A.W., Janis, C.M., & Jones, J.A. (1993) Phylogenetic analysis of covariance by computer simulation. Systematics Biology, 42, 265-292.
Garland, T.J. & Janis, C.M. (1993) Does metatarsal/femur ratio predict maximal running speed in cursorial mammals? Journal of Zoology, 229, 133-151.

exemples

```
par(mfrow=c(1,3))
phy1 <- newick2phylog(carniherbi49$tre1)
plot(phy1, clabel.leaves = 0,
 f.phylog = 2, sub ="article 1")
phy2 <- newick2phylog(carniherbi49$tre2)
plot(phy2, clabel.leaves = 0,
 f.phylog = 2, sub = "article 2")
taxo <- as.taxo(carniherbi49$taxo)
taxo <- taxo2phylog(taxo)
plot(taxo, clabel.nodes = 1.2,
 clabel.leaves = 1.2)
```



5. Inventaire forestier de la forêt de Counami

alias

CFI

description

'CFI' est un jeu de données provenant d'un inventaire forestier réalisé en Guyane française sur une surface de 12 240 hectares. 411 placettes de 0.3 hectares ont été inventoriées afin d'étudier les relations espèces/environnement et analyser les variations spatiales de la composition floristique. Les placettes sont situées sur une grille régulière dont le maillage fait 500 par 400 mètres.

format

'CFI' est une liste à trois composantes :

- *tab* : est un data frame à 411 lignes et 59 colonnes. Chaque ligne correspond à la composition floristique (mesure d'abondance) d'une placette.
- *topo* : est un facteur à 411 éléments et 12 modalités. Chaque modalité représente un type eco-topographique.
- *xy* : est un data frame à 411 lignes et deux colonnes. Chaque ligne représente les coordonnées du centre de chaque placette.

sources

http://pelissier.free.fr/diversity/CFI.html

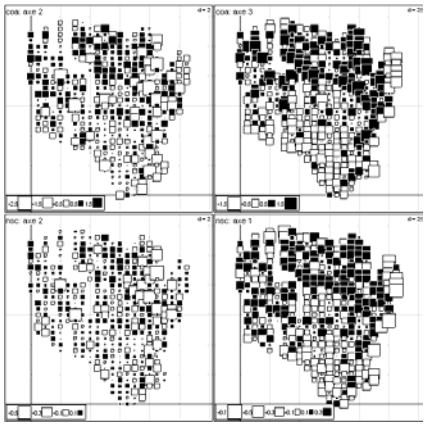
références

Couteron, P., Pélissier, R., Mapaga, D., Molino, J.F., & Teillier, L. (2002) Ecological valorisation of a management-oriented forest inventery in French Guiana. Forest Ecology and Management.

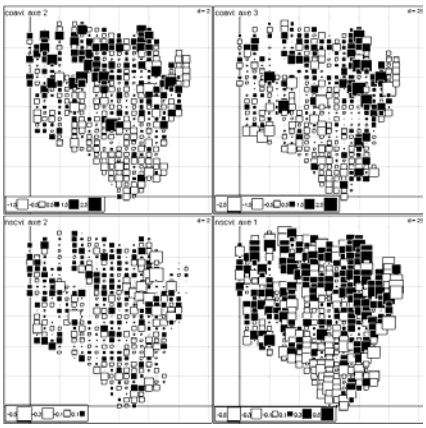
exemples

```
cfi.coa <- dudi.coa(CFI$tab)
Select the number of axes: 6
cfi.nsc <- dudi.nsc(CFI$tab)
Select the number of axes: 6
sum(cfi.nsc$eig)/59
# [1] 0.1167
# la programmation dans ade4 dans R est un peu différente de celle d'ade4
# classique: on ne retrouve l'indice de diversité de Simpson qu'à un facteur prêt
dudi.nsc.ade4 <- function (df, scannf = TRUE, nf = 2)
{
    df <- data.frame(df)
    lig <- nrow(df)
    col <- ncol(df)
    if (any(df < 0))
        stop("negative entries in table")
    if ((N <- sum(df)) == 0)
        stop("all frequencies are zero")
    row.w <- apply(df, 1, sum)/N
    col.w <- apply(df, 2, sum)/N
    df <- t(apply(df, 1, function(x) if (sum(x) == 0)
        col.w
    else x/sum(x)))
    df <- sweep(df, 2, col.w)
    df <- data.frame(df)
    X <- as.dudi(df, rep(1, col), row.w, scannf = scannf,
        nf = nf, call = match.call(), type = "nsc")
    X$N <- N
    return(X)
}
```

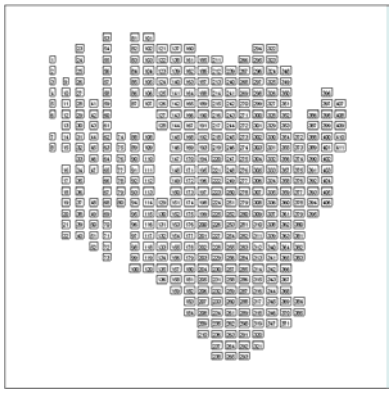
```
cfi.nsc <- dudi.nsc.ade4(CFI$tab)
Select the number of axes: 6

par(mfrow = c(2, 2))
s.value(CFI$xy, cfi.coa$li[,2],
 sub = "coa: axe 2")
s.value(CFI$xy, cfi.coa$li[,3],
 sub = "coa: axe 3")
s.value(CFI$xy, -cfi.nsc$li[,2],
 sub = "nsc: axe 2")
s.value(CFI$xy, cfi.nsc$li[,1],
 sub = "nsc: axe 1")
```

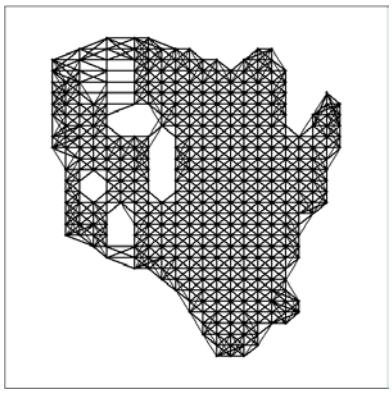


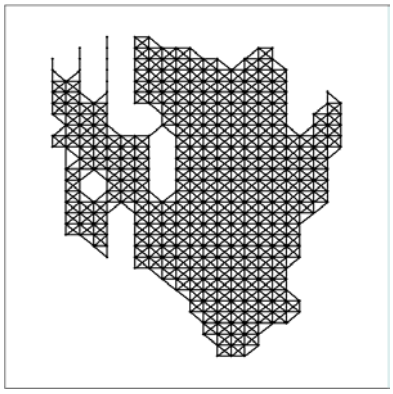
```
cfi.coa.with <- within(cfi.coa, CFI$topo)
Select the number of axes: 6
cfi.nsc.with <- within(cfi.nsc, CFI$topo)
Select the number of axes: 6
par(mfrow = c(2, 2))
s.value(CFI$xy, cfi.coa.with$li[,2],
 sub = "coavi: axe 2")
s.value(CFI$xy, cfi.coa.with$li[,3],
 sub = "coavi: axe 3")
s.value(CFI$xy, -cfi.nsc.with$li[,2],
 sub = "nscvi: axe 2")
s.value(CFI$xy, cfi.nsc.with$li[,1],
 sub = "nscvi: axe 1")
```



```
# graphe de voisinage au k-plus proche voisins
CFI.nb <- knn2nb(knearneigh(as.matrix(CFI$xy), 8))
CFI.neig <- nb2neig(CFI.nb)
CFI.listw <- nb2listw(CFI.nb)
s.label(CFI$xy, clab = 0.5, grid = FALSE, addaxes = FALSE, inc = FALSE)
s.label(CFI$xy, neig = CFI.neig, clab = 0, grid = FALSE, addaxes = FALSE, inc =
FALSE)

# graphe de voisinage defines par les distances entre voisins
CFI.nb <- dnearneigh(as.matrix(CFI$xy), 0, 0.8)
CFI.neig <- nb2neig(CFI.nb)
CFI.listw <- nb2listw(CFI.nb)
s.label(CFI$xy, neig = CFI.neig, clab = 0, grid = FALSE, addaxes = FALSE, inc =
FALSE)
```







6. Evolution de la production de 20 clémentiniers

alias

clementines

description

'clementines' est un jeu de données qui retrace la production de clémentines de 20 clémentiniers pendant 15 ans.

format

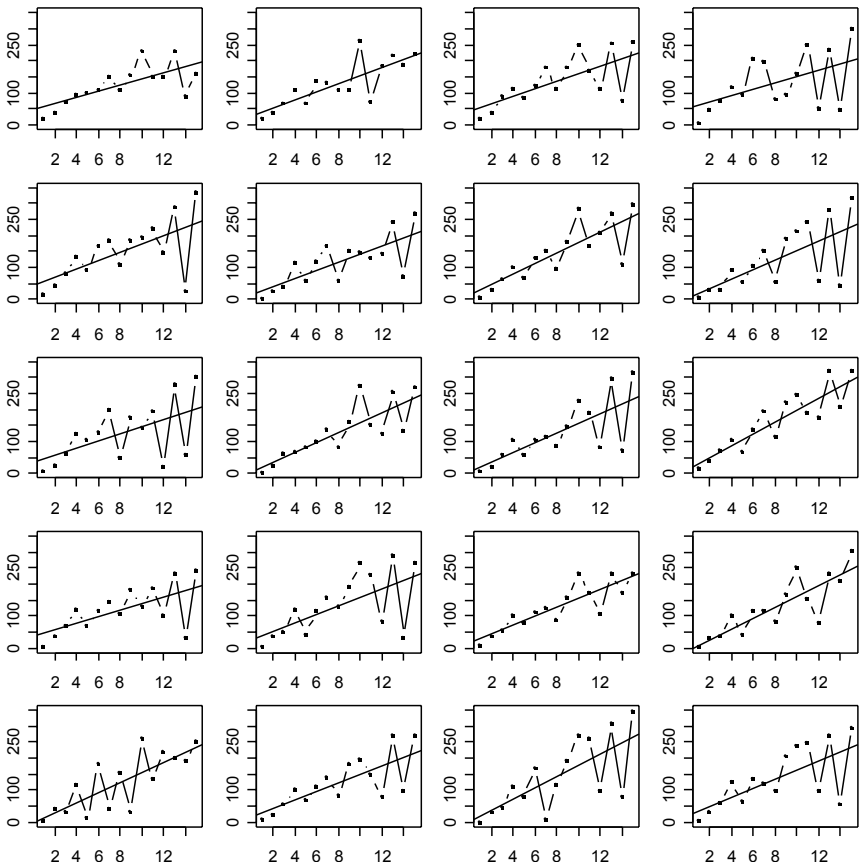
'clementines' est un data frame à 15 lignes et 20 colonnes. Chaque colonne correspond au profil évolutif de la production de chaque clémentinier. La production de clémentines est exprimée en kilogrammes.

sources

Tisné-Agostini, D. (1988) Description par analyse en composantes principales de l'évolution de la production du clémentinier en association avec 12 types de porte-greffe. Rapport technique, DEA Analyse et modélisation des systèmes biologiques, Université Lyon 1.

exemples

```
par(mfrow = c(5,4))
par(mar = c(2,2,1,1))
w0 <- 1:15
for (i in 1:20) {
plot(w0, clementines[,i], type = "b", ylim = c(0,350), pch = 20)
abline(lm(clementines[,i] ~ w0))
}
```



7. Répartition spatiale des poissons autour du lac Drouin

alias

drouin

description

'drouin' est un jeu de données sur la répartition spatiale multiéchelle des poissons autour du lac Drouin, un lac de 31 ha, d'une profondeur maximale de 22 m, situé dans la région de Lanaudière au Québec (46°09' N, 73°55' W). Huit espèces de poissons ont été dénombrées par recensement visuel en apnée à 90 sites situés le long de la rive du lac. Les recensements ont été réalisés en juin et en août 2001. Les 90 sites ont été visités trois fois de suite chaque mois. Plusieurs variables environnementales ont également été mesurées à chaque site. Pour l'instant on ne possède que le fond de carte. On a obtenu les coordonnées (en pixel) par simple digitalisation. Ce jeu de données est une bonne illustration de la variété des supports spatiaux en écologie. Il fournit un bon exemple d'utilisation des graphes circulaires.

format

'drouin' est une liste à 3 composantes :

- *xy* : est un data frame dont les 90 lignes correspondent aux coordonnées spatiales des stations.
- *neig* : est un graphe de voisinage circulaire.

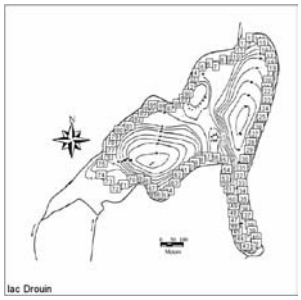
références

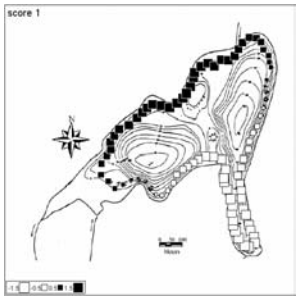
Brind'Amour, A., Boisclair, D., Legendre, P., & Borcard, D. (soumis) Multiscale spatial distribution of a littoral fish community in relation to environmental variables. Limnology and Oceanography.

Legendre, P. & Borcard, D. (2003). Quelles sont les échelles spatiales importantes dans un écosystème? In Analyse statistique de données spatiales (eds J.-J. Droesbeke, M. Lejeune & G. Saporta). Technip, Paris.

exemples

```
u <- as.character(substitute(donnees))
dirlib <- getwd()
file <- file.path(dirlib, u, "drouin.pnm")
drouin.pnm <- read.pnm(file)

s.label(drouin$xy, pixmap = drouin.pnm,
 clab = 0.75, sub = "lac Drouin",
 grid = FALSE, addaxes = FALSE)
s.label(drouin$xy, neig = drouin$neig,
 pixmap = drouin.pnm, clab = 0,
 sub = "graphe circulaire", grid = FALSE,
 addaxes = FALSE, cpoint = 0)
xy.nb <- neig2nb(drouin$neig)
plot.nb(coords = drouin$xy,
 x = xy.nb, col = "red",
 add = TRUE, points = FALSE)
points(drouin$xy, pch = 20, cex = 1.5)

# graphe circulaire
u <- orthobasis.circ(90)
s.value(drouin$xy, u[,1],
 pixmap = drouin.pnm, sub = "score 1",
 csize = 0.5, grid = FALSE,
 addaxes = FALSE, csub = 1.25)
s.value(drouin$xy, u[,89],
 pixmap = drouin.pnm,
 sub = "score 89", csize = 0.5,
 grid = FALSE, addaxes = FALSE,
 csub = 1.25)
```





8. Ordination multiéchelle

alias

gg

description

'gg' est un jeu de données simulé afin d'illustrer les problèmes associés à l'ordination multiéchelle.

format

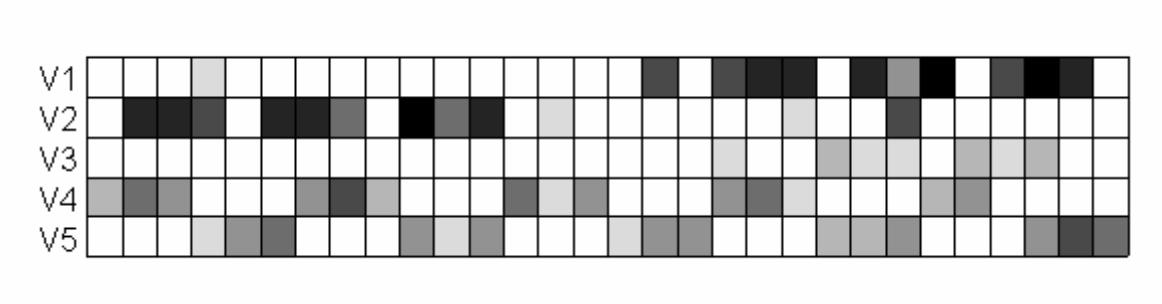
'gg' donne l'abondance de 5 espèces fictives sur un transect constitué par 30 placettes.

sources

Di Bella, G. & Jona-Lasinio, G. (1996) Including spatial contiguity information in the analysis of multispecific patterns. Environmental and Ecological Statistics, 3, 269-280.

exemples

```
table.paint(t(sqrt(gg)), cleg = 0)
```



9. Répartition spatiale du zooplancton en Guadeloupe

alias

guadeloupe

description

'guadeloupe' donne le logarithme de la biomasse de zooplancton (190 < taille < 600 µm) dans un lagon en Guadeloupe pour un transect de 51 sites irrégulièrement espacés.

format

'guadeloupe' est une liste à deux composantes :

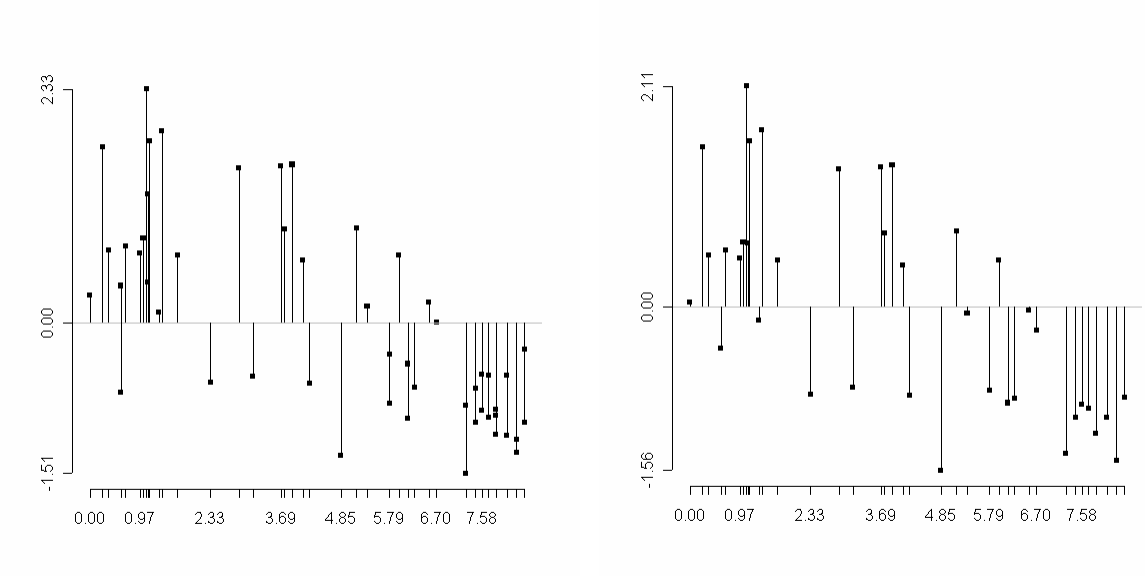
- *coord* : est un vecteur donnant la distance des sites au premier site
- *zooplankton* : est un vecteur donnant la biomasse de zooplancton

sources

Borcard, D., Legendre, P., Avois-Jacquet, C., & Tuomisto, H. (2004) Dissecting the spatial structure of ecological data at multiple scales. Ecology, 85, 1826-1832.

exemples

```
dotchart.line(guadeloupe[[2]],
 guadeloupe[[1]], csub = 0)

fac <- guadeloupe$coord
fac <- as.factor(fac)
table(fac)
fac
   0 0.24 0.36  0.6 0.69 0.97 1.03 1.09 1.12 1.15 1.33 1.39 1.69 2.33 2.88 3.15
   1    1    1    2    1    1    1    1    2    1    1    1    1    1    1    1
3.69 3.76 3.91 4.12 4.24 4.85 5.15 5.36 5.79 5.97 6.15 6.27 6.55  6.7 7.27 7.45
   1    1    1    1    1    1    1    1    2    1    2    1    1    1    2    2
7.58  7.7 7.85 8.06 8.24  8.4
   2    2    3    2    2    2
y <- guadeloupe$zooplankton
y.mean <- lapply(split(y, fac), mean)
eti <- names(y.mean)
coord <- as.numeric(eti)
biomass <- unlist(y.mean)
dotchart.line(biomass, coord, csub = 0)
```



10. Répartition spatiale de l'espèce *Adientum tomentosum* à Hunta

alias

hunta

description

'hunta' donne l'abondance de l'espèce *Adientum tomentosum* pour un transect formé de 260 quadrats de végétation de 5 m **X** 5 m, échantillonné en Amazonie péruvienne.

format

'hunta' est un vecteur de 260 éléments correspondant au nombre d'individus par cadrat.

sources

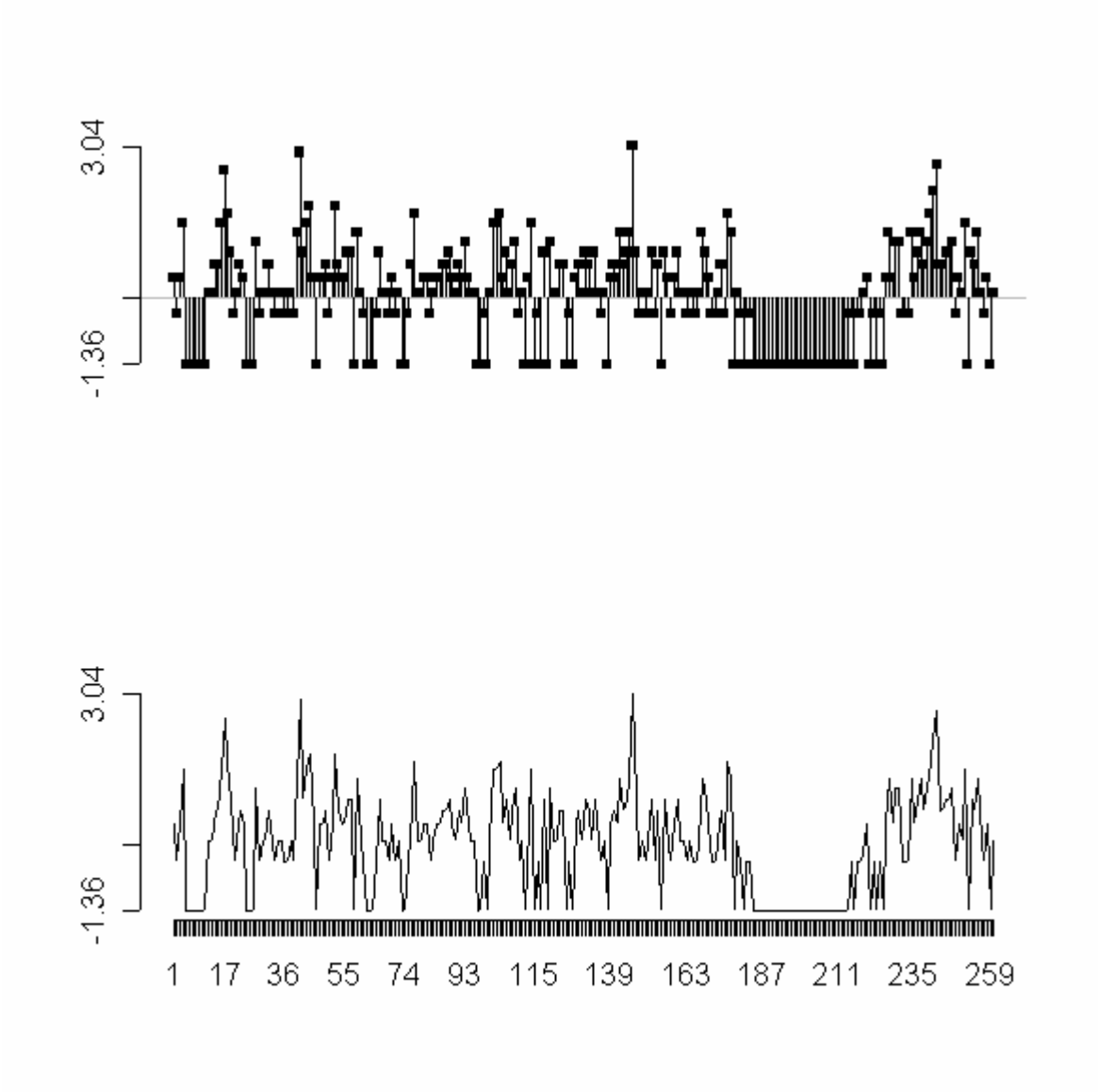
Borcard, D., Legendre, P., Avois-Jacquet, C., & Tuomisto, H. (2004) Dissecting the spatial structure of ecological data at multiple scales. Ecology, 85, 1826-1832.

références

Tuomisto, H. & Poulsen, A.D. (2000) Pteridophyte diversity and species composition in four Amazonian rain forests. Journal of Vegetation Science, 11, 383-396.

exemples

```
y <- sqrt(hunta)
par(mfrow = c(2,1))
dotchart.line(y, csub = 0, axe1 = FALSE)
dotchart.line(y, csub = 0, joining = FALSE, cdot = 0)
```



11. Données socio-économiques concernant les comtés d'Irlande

alias

irishdata

description

'irishdata' est un des jeu de données les plus connues de la statistique spatiale. Il est extrait de l'article de Geary (1954) dans lequel ce dernier a introduit pour la première fois l'indice qui porte son nom. Ce jeu de données a été ensuite réutilisé à maintes reprises, en particulier dans l'ouvrage de Cliff et Ord (1973) portant sur l'autocorrélation spatiale. Ce jeu de données contient les coordonnées géographiques et les valeurs de 12 variables socio-économiques associées aux 25 comtés d'Irlande.

format

'irishdata' est une liste à 11 composantes :

- *area* : est un data frame avec les coordonnées des 25 polygones correspondant aux limites des 25 comtés contigus.
- *county.names* : est un vecteur dont chacun des 25 éléments correspond au nom d'un comté.
- *xy* : est un data frame avec les coordonnées des centres des 25 comtés.
- *tab* : est un data frame à 25 lignes et 12 colonnes. Chaque ligne correspond à la valeur des 12 variables socio-économiques mesurées pour chaque comté. Le code des variables est le suivant : ***1-2-3*** répartition (en 1 pour 1000) des propriétés agricoles en 3 groupes d'imposition (<10 £, 10-50 £, >50 £). ***4-5-6-7*** Nombres moyens d'animaux pour 1000 acres de prairies et cultures respectivement ***4-*** vaches laitières, ***5-*** autres bestiaux, ***6-*** cochons, ***7-*** moutons. ***8-*** Pourcentage de population urbanisée (villes et villages) en 1 pour 1000. ***9-*** Nombre de voitures pour 1000 habitants. ***10-*** Nombre de licences de radio pour 1000 habitants. ***11-*** Ventes de détail moyenne par habitant en £. ***12-*** Pourcentage de célibataires parmi les hommes de 30-34 ans en 1 pour 1000.
- *contour* : est un data frame avec les coordonnées du polygone correspondant au frontière de l'Irlande.
- *link* : est une matrice de pondération de voisinage à 25 lignes et 25 colonnes dont chaque élément correspond à la longueur de la frontière commune entre deux comtés.
- *area.utm, xy.utm, link.utm, tab.utm, contour.utm* : sont les analogues des objets area, xy, link, tab, contour exprimés en coordonnées UTM (Universal Transverse Mercator). Ces objets sont directement copiés depuis l'objet 'eire' de la librairie **spdep**.

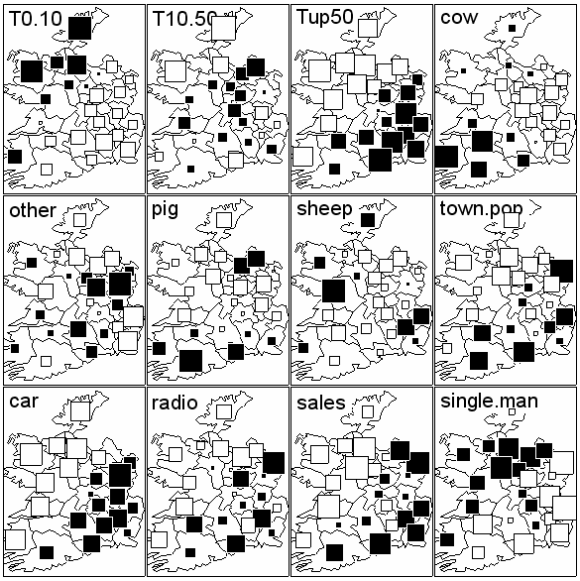
sources

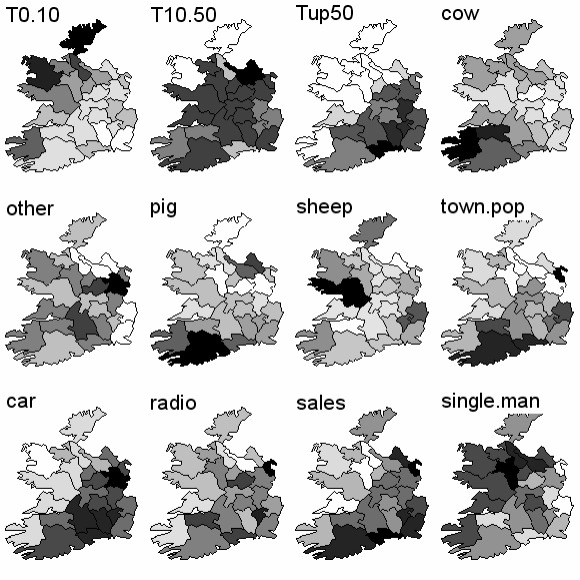
Cliff, A.D. & Ord, J.K. (1973) Spatial autocorrelation Pion, London.
Geary, R.C. (1954) The contiguity ratio and statistical mapping. The incorporated Statistician, 5, 115-145.

exemples

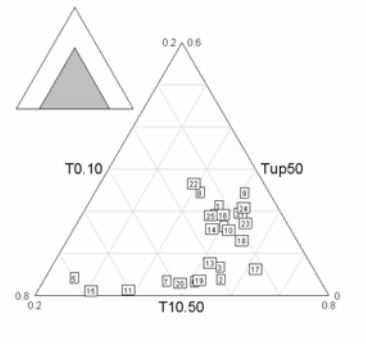
```
area.plot(irishdata$area.utm)
apply(irishdata$contour.utm, 1,
 function(x) segments(
 x[1],x[2],x[3],x[4], lwd = 3))
s.label(irishdata$area.utm[,2:3],
 cpoi = 1, clab = 0, add.p = TRUE)
area.plot(irishdata$area.utm,
 lab = row.names(irishdata$xy.utm),
 clab = 1)
```

```
par(mfrow = c(3,4))
tab <- as.data.frame(scalewt(irishdata$tab))
for (i in 1:12)
       s.value(irishdata$xy.utm, tab[,i],
        sub = names(tab)[i], csub = 3,
        csi = 2.5, cleg = 0,
        area = irishdata$area.utm,
        include = F, addax = F, grid = F)

par(mfrow = c(3,4))
for (i in 1:12)
       area.plot(irishdata$area.utm,
        val = tab[,i], sub=names(tab)[i],
        possub="topleft", csub=3, cleg=0)
```






```
# on regroupe les trois premières variables
# en 1 facteur à trois modalités

w <- irishdata$tab[,1:3]
triangle.plot(w, clab = 0.75)
kmeans(w, w[c(1,3,5), ])
$cluster
 [1] 1 2 2 1 3 2 2 1 1 1 3 1 2 1 3 1 2 1 2 2 1 1 1 1 1
$centers
  T0.10 T10.50  Tup50
1 315.6  489.5 194.79 # les plus riches
2 427.1  524.9  48.00 # les moyens
3 660.0  317.0  22.67 # les plus pauvres

# on passe d'un tableau de variables
# quantitatives à un tableau mixte

tax <- as.factor(kmeans(w,w[c(1,3,5),])$cluster)
tab <- cbind.data.frame(irishdata$tab[,-(1:3)],tax)
```



```
tab
    cow other pig sheep town.pop car radio sales single.man tax
S01  67   252  56   531      402  43   169    66         603   1
S02  99   231  97    56      173  26    56    49         734   2
S03 110   285  32   116      244  22    67    28         683   2
S04 146   256 137   148      526  38   130    66         601   1
S05 102   248  22   463      189  21    80    45         624   3
S06  69   239  44   801      281  22    87    40         691   2
S07 194   283  84   354      267  20    76    41         682   2
S08  52   290  28   184      292  40   123    54         535   1
S09  91   283  63   157      311  41    82    45         648   1
S10  69   269  54    87      267  38   121    46         672   1
S11 102   231  37    84      137  20    70    29         734   3
S12 181   277  68    36      482  32   158    53         552   1
S13  74   290  49    75      213  32   111    44         686   2
S14  69   285  55   204      630  37   200    78         510   1
S15  97   289  50   393      185  17    84    37         674   3
S16  55   351  23   252      175  49   116    53         621   1
S17  85   235 101    39      246  32    80    70         697   2
S18  55   262  50   112      356  33   110    55         652   1
S19  66   275  24   299      132  22   115    28         749   2
S20  92   266  30   205      297  24   102    42         670   2
S21 107   312  52   140      365  41   127    56         628   1
S22 122   292  96   199      564  41   164    74         546   1
S23  43   323  25   188      358  37   157    57         569   1
S24  64   219  68   288      346  34   122    66         564   1
S25  79   212  44   528      498  36   102    65         504   1

ana1 <- dudi.mix(tab)
Select the number of axes: 4

# on construit la matrice des pondérations de voisinages
# à partir de la longueur des frontières

w <- irishdata$link.utm
w1 <- t(apply(w,1,function(x) x/sum(x)))
apply(w1,1,sum)
w2 <- mat2listw(w1)
w2$style <- "W"
```

12. Données d'altimétrie laser

alias

laser

description

'laser' contient les données utilisées pour la réalisation de l'article cité en référence. En fait, il est légèrement différent de celui utilisé dans l'article car on a récupéré entre temps des limites plus précise pour les unités géomorphologiques. Toutefois, les résultats sont sensiblement les mêmes. Il est associé à l'étude de la variabilité spatiale de données d'altimétrie laser. Cette variabilité est comparée aux variations géomorphologiques sur le site de Counami, en Guyane française.

format

'article' est une liste à 3 composantes :

- *xy* : un data.frame à 264 lignes et deux colonnes. Chaque ligne définie les coordonnés spatiales d'un transect.
- *laser* : un data.frame à 264 lignes et 64 colonnes. Chaque ligne correspond au profil d'un transect laser.
- *geomorpho* : un data frame à 264 lignes et quatre colonnes. Chaque ligne définie l'appartenance de chaque transect à une unité géomorphologique selon quatre découpages successifs emboîtés. Dans l'article, c'est le niveau trois qui a été utilisé.

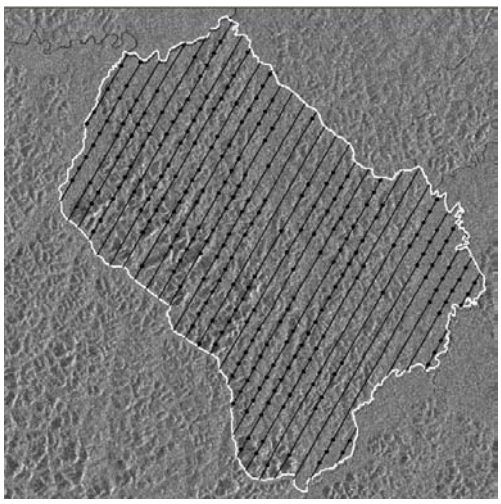
sources

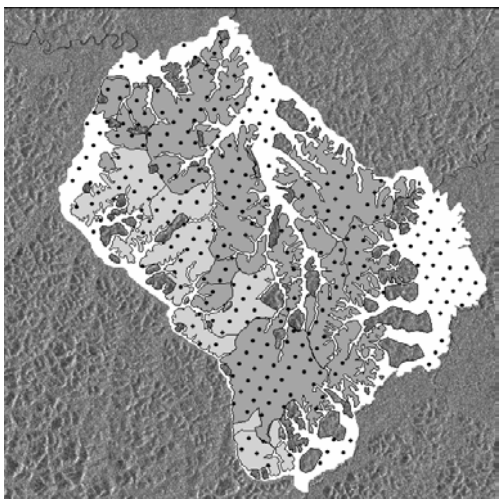
BRGM

références

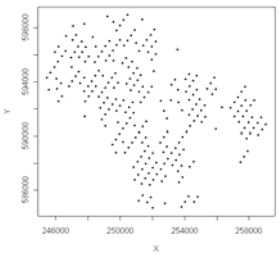
Delor, C., Perrin, J., Truffert, C., Asfirane, F., & Rossi, P. (1998) Images géophysiques dans le socle guyanais. Géochronique, 67, 7-12.
Ollier, S., Chessel, D., Couteron, P., Pélissier, R., & Thioulouse, J. (2003) Comparing and classifying one-dimensional spatial patterns: an application to laser altimeter profiles. Remote Sensing of Environment, 85, 453-462.

exemples

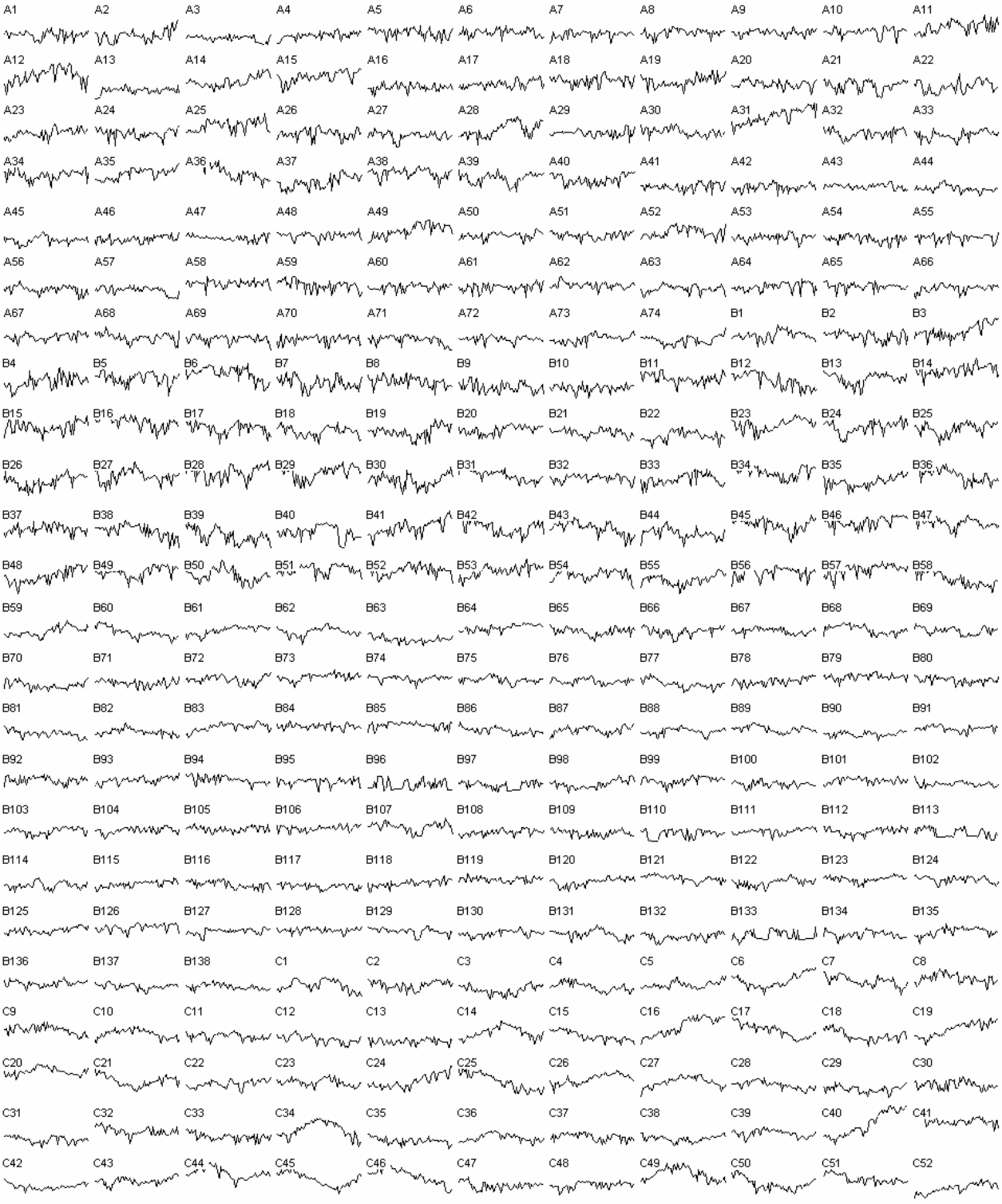




```
plot(laser$xy, pch = 20)
tab <- laser$laser
[1]  64 264
xy <- laser$xy
fac <- laser$geomorpho[,2]
table(fac)
     cd  ef  gh
 74   0 138  52

u <- split(tab, fac)
u <- lapply(u, function(x) as.data.frame(t(x)))
v <- c(74, 138, 52)
```



```
w <- LETTERS[1:3]
for(i in 1:3) names(u[[i]]) <- paste(w[i], 1:v[i], sep ="")
tab <- cbind.data.frame(u[[1]], u[[2]], u[[3]])
par(mar = c(0,0,0,0))
dotchart.line(tab[,1:132], axe1 = F, axe2 = F, cdot = 0,
 csub = 2, joining = FALSE, scaling = FALSE)
dotchart.line(tab[,133:264], axe1 = F, axe2 = F, cdot = 0,
 csub = 1.5, joining = FALSE, scaling = FALSE)
```



```
orthobas <- orthobasis.wavelet(64, "haar")
kfbs <- orthobasis2kfbs(orthobas, wavelet2level(64))
```

val <- val.kfbs(tab, kfbs)

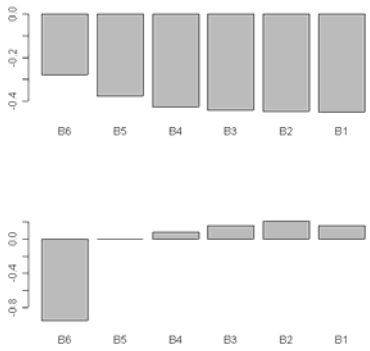
pca1 <- dudi.pca(val$vvp, center = FALSE)

Select the number of axes: 2

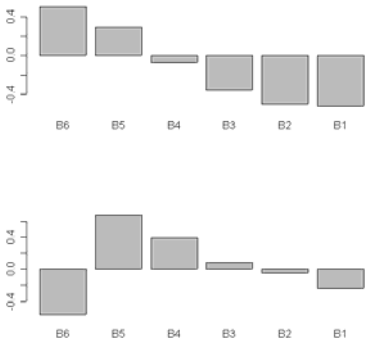
pca2 <- dudi.pca(val$vvp, center = TRUE)

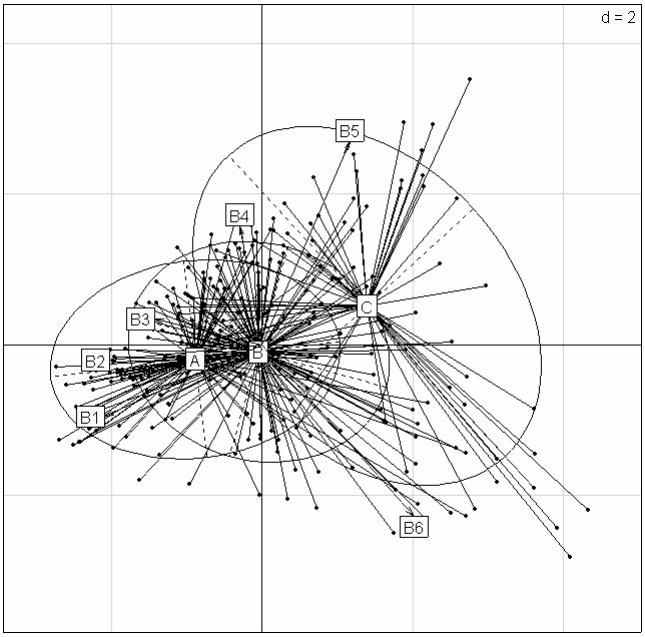
Select the number of axes: 2

```
# ACP non centrée
fac <- rep(LETTERS[1:3], v)
fac <- as.factor(fac)
boxplot(pca1$li[,1]~fac)
anova(lm(pca1$li[,1]~fac))
boxplot(pca1$li[,2]~fac)
anova(lm(pca1$li[,2]~fac))
s.label(pca1$li, clabel = 0)
s.arrow(pca1$c1, add.plot = TRUE)
s.class(pca1$li, fac, add.plot = T)
par(mfrow = c(2,1))
apply(pca1$c1, 2, barplot)
```




```
# ACP centrée
boxplot(pca2$li[,1]~fac)
anova(lm(pca2$li[,1]~fac))
boxplot(pca2$li[,2]~fac)
anova(lm(pca2$li[,2]~fac))
s.label(pca2$li, clabel = 0)
s.class(pca2$li, fac, add.plot = T)
s.arrow(pca2$c1*4, add.plot = TRUE)
par(mfrow = c(2,1))
apply(pca2$c1, 2, barplot)
```





13. Relevés phyto-écologiques dans une plaine côtière marécageuse

alias

mafragh

description

'mafragh' est un jeu de données sur la composition et le déterminisme de la végétation d'une plaine côtière marécageuse : La Mafragh (Annaba, Algérie). Il contient les coordonnées géographiques, la composition floristique et les caractéristiques environnementales pour 97 relevés phyto-écologiques.

format

'mafragh' est une liste à 7 composantes :

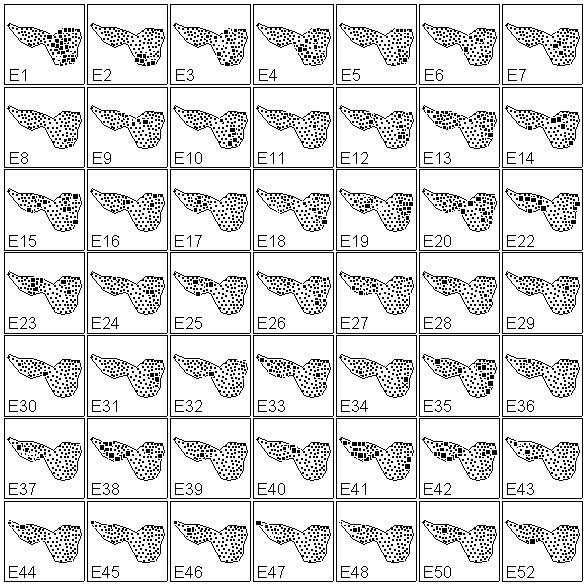
- *xy* : est un data frame contenant les coordonnées spatiales des 97 relevés
- *flo* : est un data frame à 97 lignes et 56 colonnes. Chaque ligne correspond à la composition floristique de chaque relevé
- *espnames* : est un vecteur donc chaque élément correspond au nom d'une espèce
- *neig* : est un graphe de voisinage associé aux 97 relevés
- *mil* : est un data frame à 97 lignes et 11 colonnes. Chaque ligne correspond aux caractéristiques environnementales de chaque relevé
- *partition* : est un facteur à 7 modalités
- *area* : est un data frame contenant les coordonnées des polygones du pavage de Voronoï associé aux relevés

sources

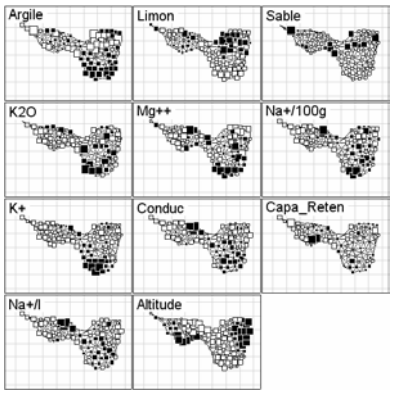
de Belair, G. & Bencheikh-Lehocine, M. (1987) Composition et déterminisme de la végétation d'une plaine côtière marécageuse : La Mafragh (Annaba, Algérie). Bulletin d'Ecologie, 18, 393-407.

exemples

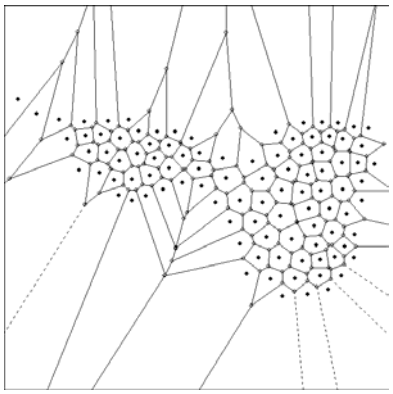
```
xy <- mafragh$xy
flo <- mafragh$flo
flo <- mafragh$flo[,
 apply(mafragh$flo,2,sum)>3]
flonames <- mafragh$espnames[
 apply(mafragh$flo,2,sum)>3]
# on conserve 97 sites et 49 espèces
# dont l'abondance vaut plus de 3

carto.mafragh <- function (x) {
       w1 <-c(253,208,144,75,23,13,81,
        140,191,208,237,261,301,351,377,
        378,399,387,300,252)
       w2 <- c(156,180,199,199,222,209,
        125,96,115,111,53,9,1,10,51,93,
        147,190,195,156)
       w3 <- as.factor(rep("1",20))
       wessai <- cbind.data.frame(w3,w1,w2)
       par(mfrow = c(7,7))
       par(mar = c(2, 0.5,  0.5, 0.5))
       for (i in 1:49) {
              area.plot(wessai,
               sub = names(x)[i],
               possub = "bottomleft",
               csub = 2.5)
              s.label(xy, add.p = T,
               cpoi = 0.25, clab = 0)
              s.value(xy, x[,i],
               add.p = T, csi = 1,
               cleg = 0)}
}

carto.mafragh(flo)
```

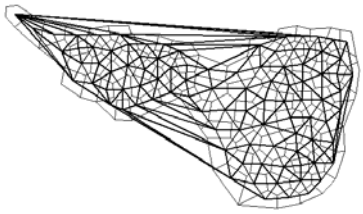


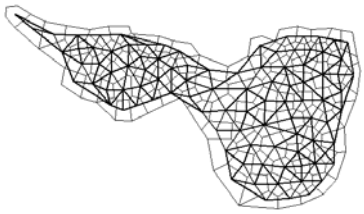
```
tab0 <- (as.data.frame(
 scalewt(mafragh$mil)))
par(mfrow = c(4,3))
for (k in 1:11)
       s.value(mafragh$xy,tab0[,k],
        neig = mafragh$neig, cleg = 0,
        cgrid = 0, addax = F,
        sub = names(tab0)[k], csub = 3)
```




```
par(mar = c(0, 0, 0, 0))
plot(xy, asp = 1, pch = 20, cex = 1.5)
plot(voronoi.mosaic(xy), add = TRUE)
maf1 <- tri2nb(mafragh$xy)
area.plot(mafragh$area,graph = nb2neig(maf1))

# utiliser la fonction 'edit.nb'
# pour éliminer les arêtes non pertinentes

area.plot(mafragh$area,graph = mafragh$neig)
```







14. Traits et phylogénie de poissons marins

alias

mjrochet

description

'mjrochet' donne 7 traits biologiques de 49 espèces de poissons téléostéens ainsi qu'un arbre phylogénétique des espèces.

format

'mjrochet' est une liste à deux composantes :

- *tre* : est un arbre phylogénétique au format newick (49 espèces)
- *tab* : est un data frame à 49 lignes (espèces) et 7 colonnes (traits biologiques) :

1. " tm "âge de maturité sexuelles (années)
2. " lm "longueur à la maturité sexuelle (cm
3. " l05 " longueur au temps de 5% de survivants (cm
4. " t05 " temps écoulé entre la maturité sexuelle et le temps de 5% de survivants
5. " fb "mesure de l'accroissement de la fécondité avec la taille des femelles
6. " fm "fécondité à la maturité
7. " egg " volume des œufs

sources

http://www.ifremer.fr/maerha/plan.htm

Summary of data - Clupeiformes : http://www.ifremer.fr/maerha/clupe.html
Summary of data - Argentiniformes : http://www.ifremer.fr/maerha/argentin.html
Summary of data - Salmoniformes : http://www.ifremer.fr/maerha/salmon.html
Summary of data - Gadiformes : http://www.ifremer.fr/maerha/gadi.html
Summary of data - Lophiiformes : http://www.ifremer.fr/maerha/loph.html
Summary of data - Atheriniformes : http://www.ifremer.fr/maerha/ather.html
Summary of data - Perciformes : http://www.ifremer.fr/maerha/perci.html
Summary of data - Pleuronectiformes : http://www.ifremer.fr/maerha/pleuro.html
Summary of data - Scorpaeniformes : http://www.ifremer.fr/maerha/scorpa.html

On a moyenné les données relatives aux différentes populations de la même espèce pour ne garder que le problème phylogénétique

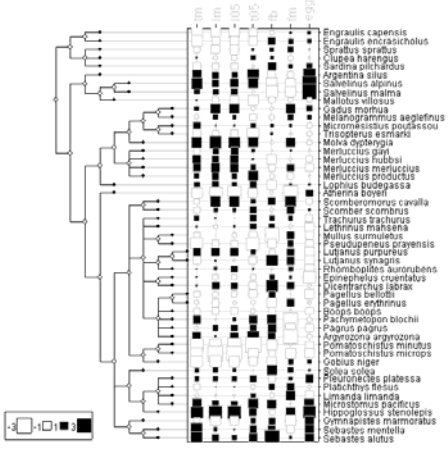
Phylogénie en format graphique :
http://www.ifremer.fr/maerha/life_history.html

références

Cornillon, P.-A. (1998) Prise en compte de proximités en analyse factorielle et comparative. Thèse, Ecole Nationale Supérieure Agronomique, Montpellier.
Cornillon, P.-A., Pontier, D., & Rochet, M.J. (2000) Autoregressive models for estimating phylogenetic and environmental effects: accounting for within-species variations. Journal of Theoretical Biology, 202, 247-256.
Rochet, M.J., Cornillon, P.-A., Sabatier, R., & Pontier, D. (2000) Comparative analysis of phylogenic and fishing effects in life history patterns of teleos fishes. Oïkos, 91, 255-270.

exemples

```
mjrochet.phy <- newick2phylog(mjrochet$tre)
tab <- log((mjrochet$tab))
tab0 <- data.frame(scalewt(tab))
table.phylog(tab0,mjrochet.phy,
 csi = 2, clabel.row = 0.75)
```



15. Répartition spatiale de l'espèce *Adientum tomentosum* à Nauta

alias

nauta

description

'nauta' donne l'abondance de l'espèce *Adientum tomentosum* pour transect formé de 260 quadrats de végétation de 5 m **X** 5 m, échantillonné en Amazonie péruvienne.

format

'nauta' est un vecteur de 260 éléments correspondant au nombre d'individus par quadrat.

sources

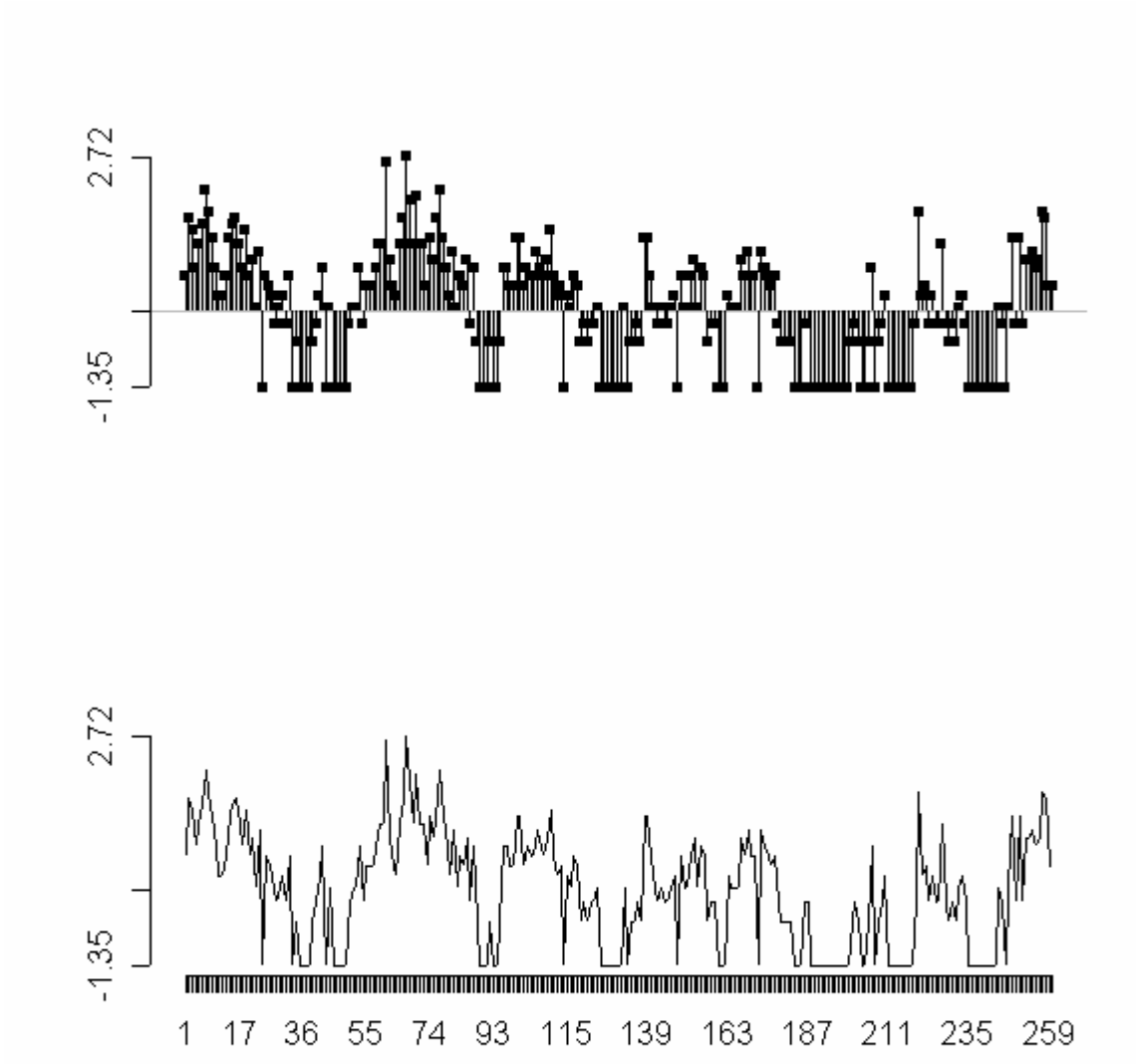
Borcard, D., Legendre, P., Avois-Jacquet, C., & Tuomisto, H. (2004) Dissecting the spatial structure of ecological data at multiple scales. Ecology, 85, 1826-1832.

références

Tuomisto, H. & Poulsen, A.D. (2000) Pteridophyte diversity and species composition in four Amazonian rain forests. Journal of Vegetation Science, 11, 383-396.

exemples

```
y <- sqrt(nauta)
par(mfrow = c(2,1))
dotchart.line(y, csub = 0, axe1 = FALSE)
dotchart.line(y, csub = 0, joining = FALSE, cdot = 0)
```



16. Exemples de phylogénies au format 'Newick'

alias

newick.eg

description

'newick.eg' fournit une série d'exemples d'arbres phylogénétiques au format 'Newick'

format

'newick.eg' est une liste à 14 composantes. Chaque élément est un vecteur de chaînes de caractères contenant un arbre au format 'Newick'

sources

1 à 7 : exemples de base dans :

http://evolution.genetics.washington.edu/phylip/newicktree.html

8 et 9 : communication personnelle de C. Gimaret_Carpentier

10 :
Treezilla Data Sets : (500-taxon rbcL Analyses) dans :

http://www.cis.upenn.edu/~krice/treezilla/
http://www.cis.upenn.edu/~krice/treezilla/ann.06.contree

11 et 12 : treeA and treeB p.96 in
Bauwens, D., and R. Díaz-Uriarte. 1997. Covariation of life-history traits in lacertid lizards: a comparative study. American Naturalist **149**:91-111.

13 Cheverud, J., and M. M. Dow. 1985. An autocorrelation analysis of genetic variation due to lineal fission in social groups of rhesus macaques. American Journal of Physical Anthropology **67**:113-122.

14 exemple fictif dans Martins, E. P., and T. F. Hansen. 1997. Phylogenies and the comparative method: a general approach to incorporating phylogenetic information into the analysis of interspecific data. The American Naturalist **149**:646-667.

références

Treezilla Data Sets :

Chase, M.W., Soltis, D.E., Olmstead, R.G., Morgan, D., Les, D.H., Mishler, B.D., Duvall, M.R., Price, R.Q., Hills, H.G., Qiu, Y.-L., Kron, K.A., Rettig, J.H., Conti, E., Palmer, J.D., Manhart, J.R., Sytsma, K.J., Michaels, H.J., Kress, W.J., Karol, K.G., Clark, W.D., Hedren, M., Gaut, B.S., Jansen, R.K., Kim, K.-J., Wimpee, C.F., Smith, J.F., Furnier, G.R., Strauss, S.H., Xiang, Q.-Y., Plunkett, G.M., Soltis, P.S., Swensen, S., Williams, S.E., Gadek, P.A., Quinn, C.J., Eguiarte, L.E., Golenberg, E., Learn, G.H.J., Graham, S.W., Barrett, S.C.H., Dayanandan, S., & Albert, V. (1993) Phylogenetics of seed plants: an analysis of nucleotide sequences from the plastid gene rbcL. Annals of the Missouri Botanical Garden, 80, 528-580.

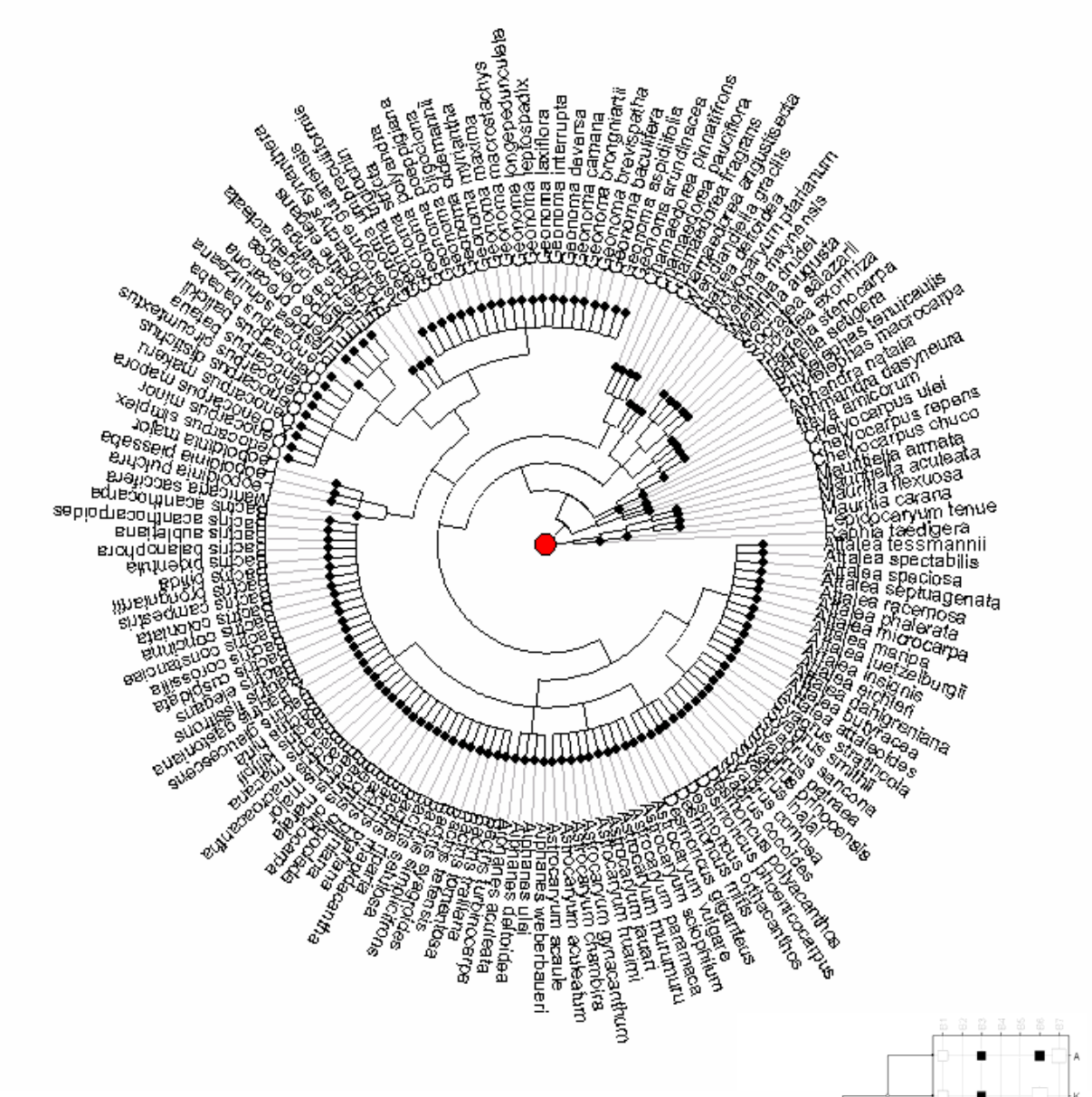
Usage de 13 dans Rohlf, F.J. (2001) Comparative methods for the analysis of continuous variables: geometric interpretations. Evolution, 55, 2143-2160.

exemples

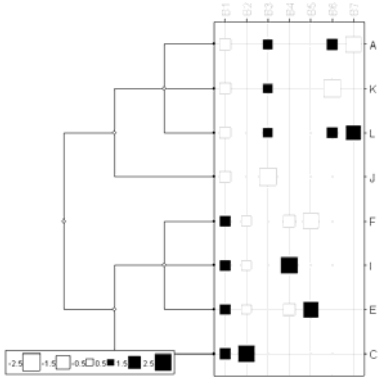
```
newick2phylog(newick.eg[[11]])
Phylogenetic tree with 18 leaves and 17 nodes
$class: phylog
$call: newick2phylog(x.tre = newick.eg[[11]])
$tre: ((Sa,Sh)I1,((((Tl,(Mc,My)...Ls,Lv)I13)I14)I15)I16)Root;

        class   length content
$leaves numeric 18     length of the first preceeding adjacent edge
$nodes  numeric 17     length of the first preceeding adjacent edge
$parts  list    17     subsets of descendant nodes
$paths  list    35     path from root to node or leave
```

```
$droot  numeric 35     distance to root

          class       dim    content
$Wmat     matrix      18-18 W matrix : root to the closest ancestor
$Wdist    dist        153   Nodal distances
$Wvalues  numeric     17    Eigen values of QWQ/sum(Q)
$Wscores  data.frame  18-17 Eigen vectors of QWQ '1/n' normed
$Amat     matrix      18-18 Topological proximity matrix A
$Avalues  numeric     17    Eigen values of QAQ matrix
$Adim     integer     1     number of positive eigen values of QAQ
$Ascores  data.frame  18-17 Eigen vectors of QAQ '1/n' normed
$Aparam   data.frame  17-3  Topological indices for nodes
$Bindica  data.frame  18-17 class indicator from nodes
$Bscores  data.frame  18-17 Topological orthonormal basis '1/n' normed
$Blabels  character   17    Nodes labelling from orthonormal basis
```

```
radial.phylog(newick2phylog(newick.eg[[8]]),
 circ = 1, clabel.l = 0.75)
```




```
che.phy <- newick2phylog(
 newick.eg[[13]])
table.phylog(che.phy$Bscores,
```



```
che.phy, csi = 2)
```

17. Variabilité spatiale dans une communauté d'Oribates

alias

oribatid

description

'oribatid' est un jeu de données classique en écologie qui contient un tableau faunistique avec 35 espèces et 70 éléments d'échantillonnage. Les éléments d'échantillonnage sont des carottes de sol de 5cm de diamètre et 10 cm de profondeur. Il contient également les coordonnées des 70 éléments d'échantillonnage ainsi qu'un tableau renseignant sur 5 variables environnementales.

format

'oribatid' est une liste à trois composantes :

- *fau* : est un data frame à 70 lignes (carottes) et 35 colonnes (espèces) Chaque case du tableau correspond à l'abondance d'un espèce dans une carotte.
- *envir* : est un data frame à 70 lignes (carottes) et 5 colonnes (variables environnementales). Les variables environnementales sont le ***substrat*** (facteur à 7 modalités sph1 (Sphaignes, groupe d'espèces 1), sph2 (Sphaignes, groupe d'espèces 2), sph3 (Sphaignes, groupe d'espèces 3), sph4 (Sphaignes, groupe d'espèces 4), litter (litière), peat (tourbe nue) et inter (interfaces))) ; **la densité en buisson** (facteur à 3 modalités none (aucun), few (un peu) et many (beaucoup).) ; **la microtopographie** (facteur à 2 modalités blanket (replat), hummock (butte)) ; **la densité du substrat** en $g.L^{-1}$ de matière sèche non comprimée ; **le contenu en eau** du substrat en $g.L^{-1}$.
- *xy* : est un data frame à 70 lignes et 2 colonnes correspondant aux coordonnées spatiales des éléments d'échantillonnage.

sources

http://www.fas.umontreal.ca/biol/casgrain/fr/labo/oribates.html

références

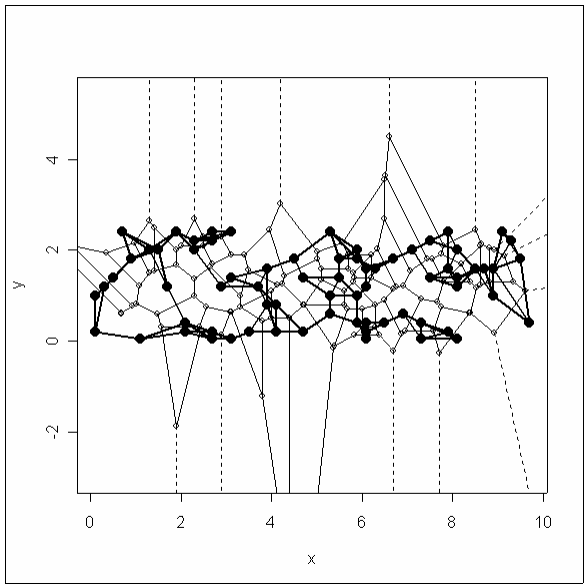
Borcard, D. & Legendre, P. (1994) Environmental control and spatial structure in ecological communities: an example using oribatid mites (Acari, Oribatei). Environmental and Ecological Statistics, 1, 37-61.

Borcard, D., Legendre, P., & Drapeau, P. (1992) Partialling out the spatial component of ecological variation. Ecology, 73, 1045-1055.

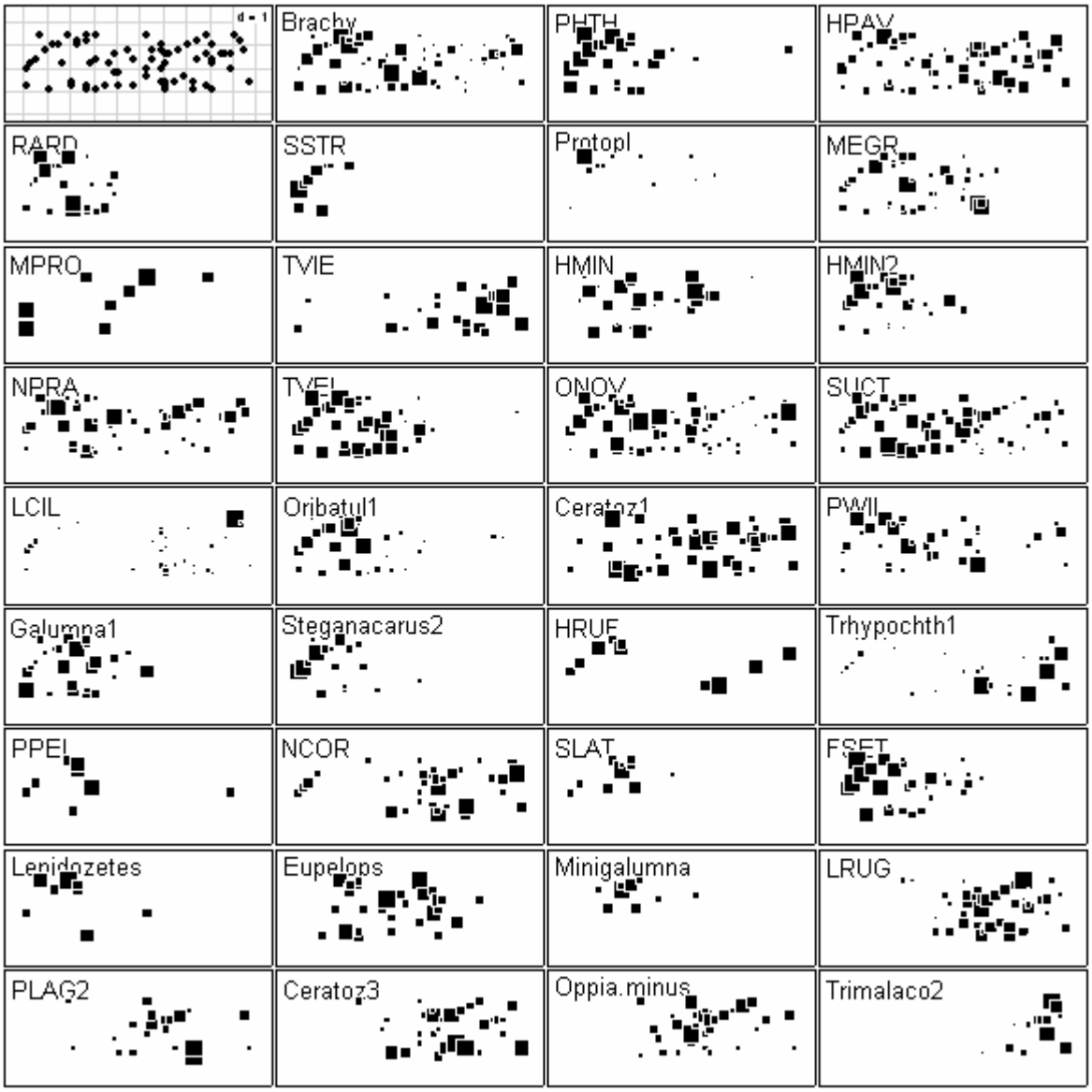
Wagner, H.H. (2004) Direct multi-scale ordination with canonical correspondence analysis. Ecology, 85, 342-351.

exemples

```
ori.xy <- oribatid$xy[,c(2,1)]
names(ori.xy) <- c("x","y")
plot(ori.xy,pch = 20, cex = 2, asp = 1)
plot(voronoi.mosaic(ori.xy), add = TRUE)
ori.neig <- nb2neig(knn2nb(
 knearneigh(as.matrix(ori.xy), 3)))
s.label(ori.xy, add.p = TRUE,
 neig = ori.neig, clab = 0)
```



```
par(mfrow=c(9,4))
s.label(ori.xy, clab=0, incl=F, addax=F, cpoi=2)
for(j in 1:35)
       s.value(ori.xy, oribatid$fau[,j],
        cleg=0, incl=F, grid = F, addax=F,
        sub = names(oribatid$fau)[j], csub = 2)
```




```
ori.listw.3nn <- nb2listw(knn2nb(
 knearneigh(as.matrix(ori.xy),3)))
ori.log <- log(oribatid$fau + 1)
ori.pca <- dudi.pca(ori.log, scale = F)
Select the number of axes: 4
ori.mix <- dudi.mix(oribatid$envir)
Select the number of axes: 3
```

18. Phylogénie et traits biologiques des procellariiformes

alias

procella

description

'procella' donne 6 traits biologiques de 19 espèces d'oiseaux du genre procellariiformes ainsi qu'un arbre phylogénétique des espèces.

format

'procella' est une liste à deux composantes :

- tre : est un arbre phylogénétique au format newick (19 espèces)
- traits : est un data frame à 19 lignes (espèces) et 6 colonnes (traits biologiques) :

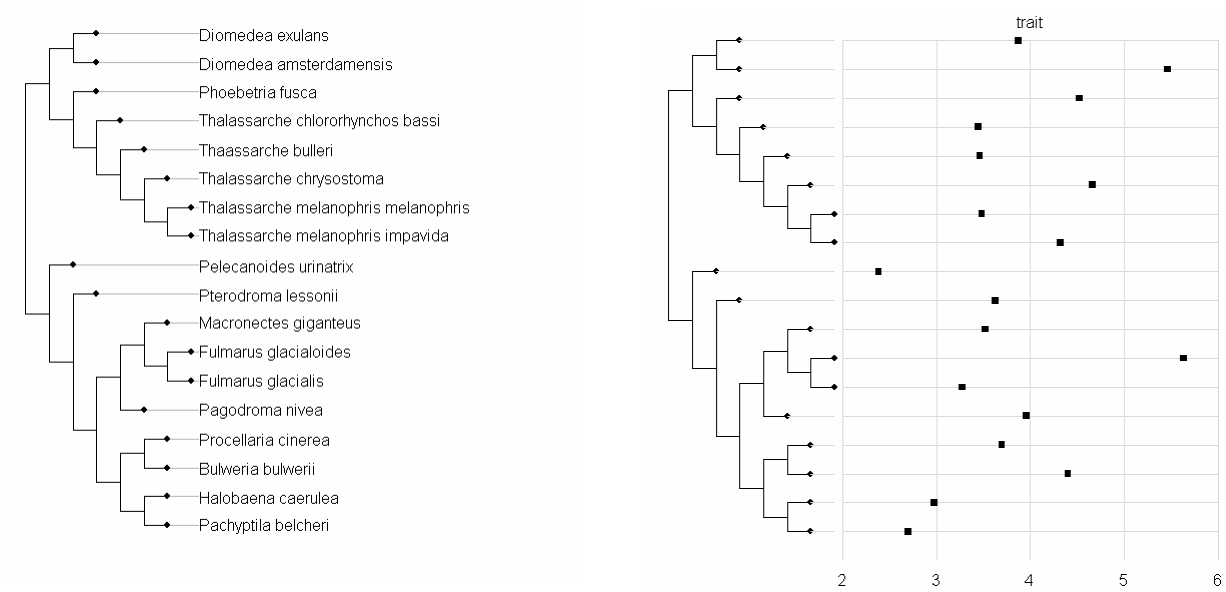
1. " site.fid " % de fidélité au site
2. " mate.fid " % de fidélité au nid
3. " mass " masse des adultes (g)
4. " ALE " espérance de vie des adultes
5. " BF " fréquence de reproduction
6. " col.size " taille des colonnies

sources

Bried, J., Pontier, D., & Jouventin, P. (2003) Mate fidelity in monogamus birds: a re-examination of the Procellariiformes. Animal Behaviour, 65, 235-246.

exemples

```
procella.phy <- newick2phylog(procella.phy$tre)
procella.trait<-procella$traits$ALE
procella.trait<-procella$traits$ALE[!is.na(procella.trait)]
procella.trait <- sqrt(procella.trait)
plot.phylog(procella.phy, f = 0.3)
tab <- data.frame(procella.trait)
names(tab) <- c("trait")
dotchart.phylog(procella.phy,tab, f = 0.3,
 scaling = F, ranging = F, joining = F)
```



19. Répartition spatiale des espèces dans un "fourré tigré" au Burkina Faso

alias

savanne

description

'savanne' donne l'abondance de 5 espèces d'un peuplement ligneux dans un fourré tigré (***B**oscia **s**enegalensis*, ***G**ardenia **s**okotensis*, ***G**rewia **b**icolor*, ***G**uiera **se**negalensis*, ***P**terocarpus **l**ucens*). Ce fourré tigré est bien structuré. Il a été sélectionné sur un glacis peu incliné (pente 0.7%) d'environ 3 km de long et 2 km de large, reliant un sommet cuirassé à un bas fond. L'échantillonnage retenu se base sur 224 placettes de 12.5*12.5m (4 transects de 56 placettes).

Format

'savanne' est data frame à 20 colonnes et 56 lignes. Chaque colonne représente l'abondance d'une des 5 espèces dans un des 4 transects.

sources

Pierre Couteron

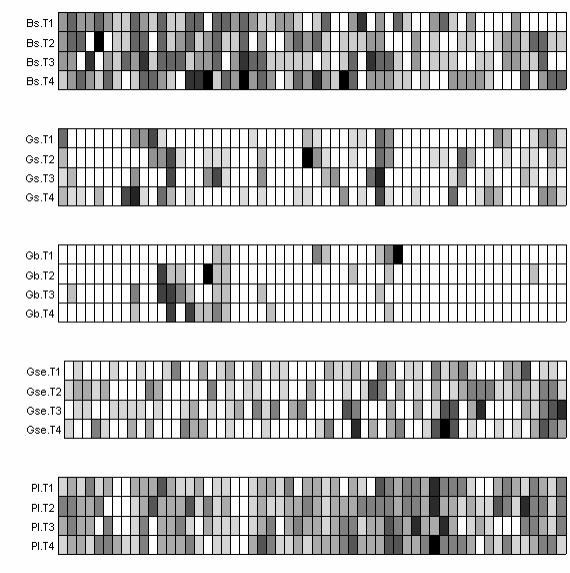
références

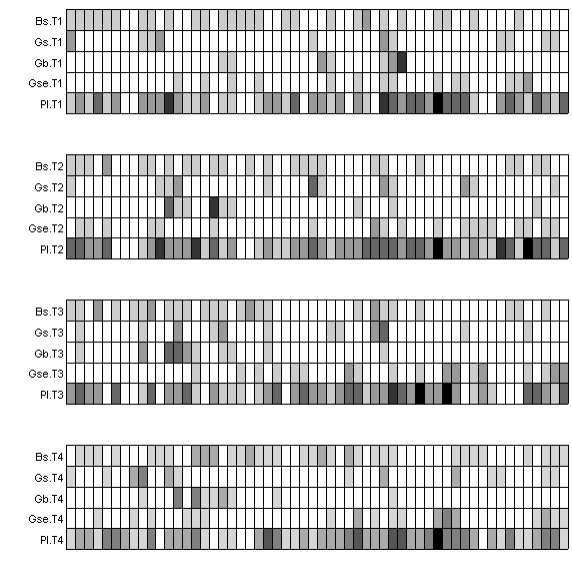
Couteron, P. & Kokou, K. (1997) Woody vegetation spatial patterns in a semi-arid savanna of Burkina Faso, West Africa. Pl0, 132, 221-227.

Couteron, P., Mahamane, A., & Ouedraogo, P. (1996) Analyse de la structure de peuplements ligneux dans un "fourré tigré" au nord Yatenga (Burkina Faso). Etat actuel et conséquences évolutives. Annales des Sciences Forestières, 53, 867-884.

exemples

```
# représentation par espèces
par(mfrow = c(5,1))
fac <- as.factor(rep(1:5, 4))
u <- split(as.data.frame(
 t(sqrt(savanne))), fac)
lapply(u, function(x) table.paint(
 x, cleg = 0, clabel.row = 1.5,
 clabel.col = 0))


# représentation par transects
par(mfrow = c(4,1))
fac <- as.factor(rep(1:4, rep(5, 4)))
u <- split(as.data.frame(
 t(sqrt(savanne))), fac)
lapply(u, function(x) table.paint(
 x, cleg = 0, clabel.row = 1.5,
 clabel.col = 0))
```





20. Transect de vegetation

alias

steppe

description

'steppe' est un jeu de données correspondant au relevé floristique de 37 espèces sur 512 sites.

format

'steppe' est une liste à deux composantes :

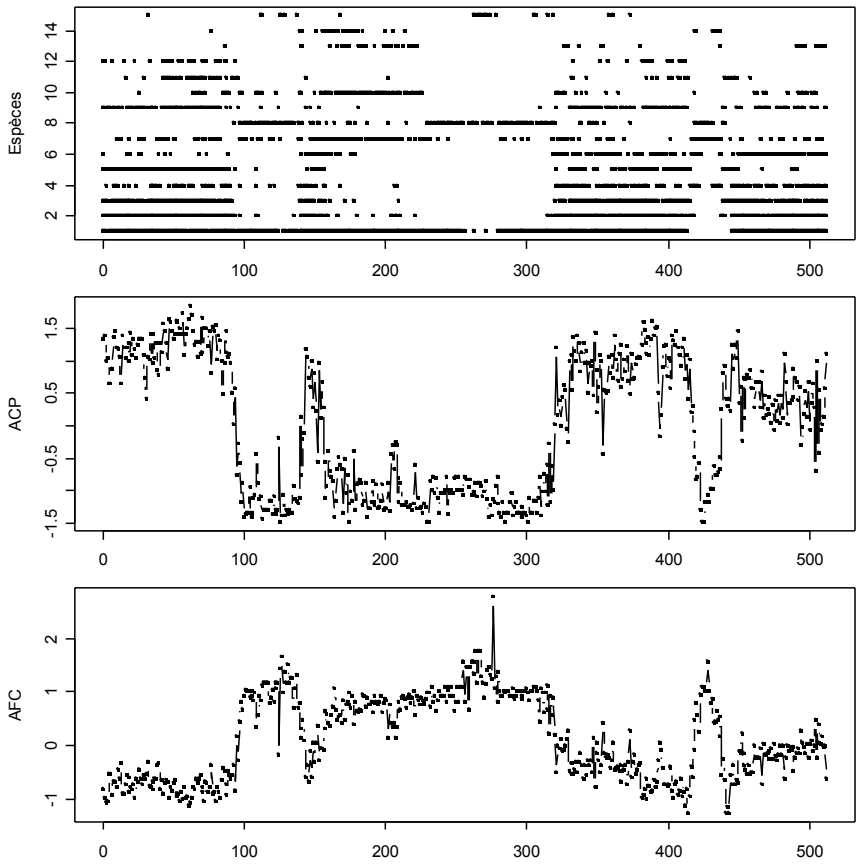
- *tab* : est un data frame à 512 lignes (sites) et 37 variables en absence-présence (espèces).
- *esp.names* : est un vecetur dont chaque élément correspond au nom d'une espèce.

sources

Estève, J. (1978). Les méthodes d'ordination : éléments pour une discussion. In Biométrie et Ecologie (eds J.M. Legay & R. Tomassone), pp. 223-250. Société Française de Biométrie, Paris.

exemples

```
par(mfrow = c(3,1))
par(mar=c(2.1,4.1,1.1,1.1))
w1 <- col(as.matrix(steppe$tab[,1:15]))
w1 <- as.numeric(w1[steppe$tab[,1:15] > 0])
w2 <- row(as.matrix(steppe$tab[,1:15]))
w2 <- as.numeric(w2[steppe$tab[,1:15] > 0])
plot(w2, w1, pch = 20, ylab="Espèces", xlab="", cex=0.75)
plot(dudi.pca(steppe$tab, scan = FALSE, scale = FALSE)$li[,1], pch = 20, ylab =
"ACP", xlab = "", type = "b", cex=0.75)
plot(dudi.coa(steppe$tab, scan = FALSE)$li[,1], pch = 20, ylab = "AFC", xlab = "",
type = "b", cex=0.75)
```



21. Températures mensuelles moyennes de 30 villes françaises

alias

t3012

description

't3012' est un jeu de données qui regroupe les températures moyennes mesurées chaque mois dans 30 villes différentes. Les coordonnées spatiales sont également fournies, ce qui permet d'envisager une typologie de carte de la température en fonction de la date. On pourrait également étudier la variabilité des profils annuels en fonction des villes.

format

't3012' est une liste à 3 composantes :

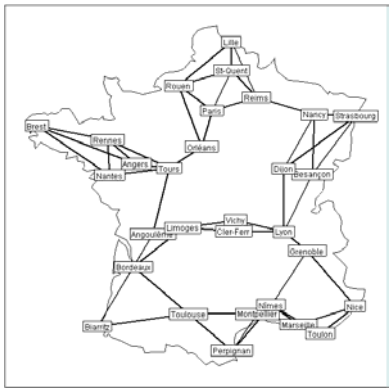
- *xy* : est un data frame à 30 lignes et 2 colonnes. Chaque ligne contient les coordonnées spatiales de chacune des 30 villes.
- *temp* : est un data frame à 30 lignes et 12 colonnes. Chaque colonne représente les températures moyennes pour un des douze mois de l'année considéré dans chacune des 30 villes.
- *contour* : est un data frame qui contient les coordonnées du polygone de contour correspondant aux frontières de la France.

sources

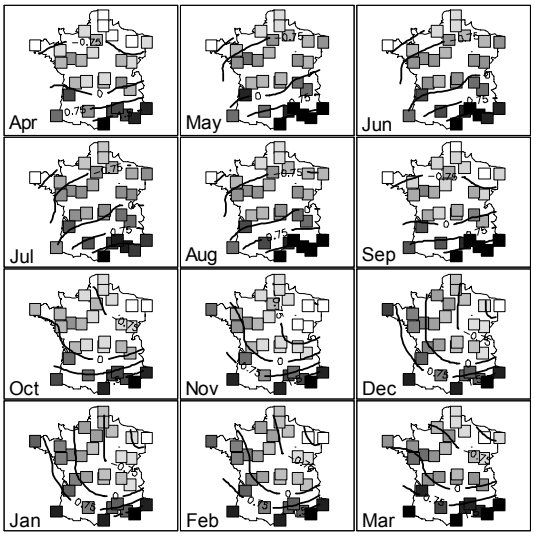
Besse, P. (1979) Etude descriptive d'un processus ; approximation, interpolation. Thèse de 3ème cycle, Université Paul Sabatier, Toulouse.

exemples

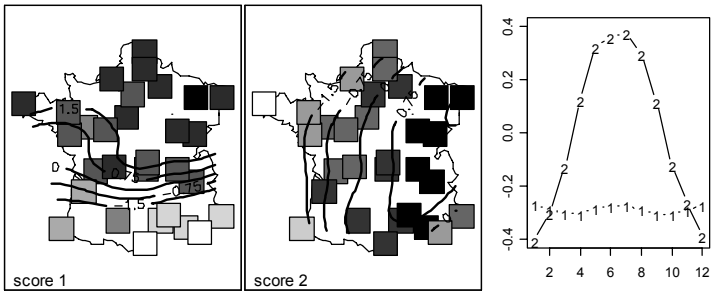
```
t3012.nb <- knn2nb(knearneigh(
 as.matrix(t3012$xy),3))
t3012.neig <- nb2neig(t3012.nb)
t3012.listw <- nb2listw(t3012.nb)
s.label(t3012$xy, clab = 0.75,
 cpoi = 0, neig = t3012.neig,
 contour = t3012$contour,
 grid = FALSE, addaxes = FALSE,
 inc = FALSE)
```




```
"cdn" <- function(xy, z, poly4, span = 0.25, sub){
par(mar=c(0,0,0,0))
w <- cbind.data.frame(xy,z)
lo <- loess(z~x+y, data = w, span = span)
xg <- seq(min(xy[,1]),max(xy[,1]),le=nrow(xy))
yg <- seq(min(xy[,2]),max(xy[,2]),le=nrow(xy))
gr <- expand.grid(xg, yg)
names(gr) <- names(xy)
polyin <- xy[chull(xy),]
grin <- inpip(gr,polyin)
mod <- predict(lo,newdata=gr)
mod[-grin] <- NA
mod <- matrix(mod,nrow(xy),nrow(xy))
s.label(xy, include.ori = F, addax = F,
 cpoi = 1, clab = 0, contour  =poly4,
 cgrid = 0,grid = F, sub = sub, csub = 3)
s.value(xy,z,add.p=T,cleg=0, meth="greylevel")
contour(xg, yg, mod, add = TRUE,
 labcex = 0.75, lwd = 2, nlevels = 5,
levels = c(-1.5,-0.75,0,0.75,1.5))
}
par(mfrow=c(4,3))
for (j in c(4:12,1:3))
      cdn(t3012$xy,scalewt(t3012$temp[,j]),
       t3012$contour,0.7,names(t3012$temp)[j])
```



```
t3012.pca <- dudi.pca(t3012$temp)
Select the number of axes: 2
par(mfrow = c(1, 3))
cdn(t3012$xy, t3012.pca$li[,1],
 t3012$contour, 0.7, « score 1 »)
cdn(t3012$xy, t3012.pca$li[,2],
 t3012$contour, 0.7, « score 2 »)
par(mar = c(3,3,1,1))
plot(1:12,t3012.pca$c1[,1],
 ylim = c(-0.4,0.4), type = "b",
 pch="1", xlab = « mois »,
 ylab = « »)
points(1:12,t3012.pca$c1[,2],
 type = "b", pch = "2")
```



22. Exemples de taxonomie

alias

taxo.eg

description

'taxo.eg' fournit deux exemples de taxonomie.

format

'taxo.eg' est une liste à deux composantes :

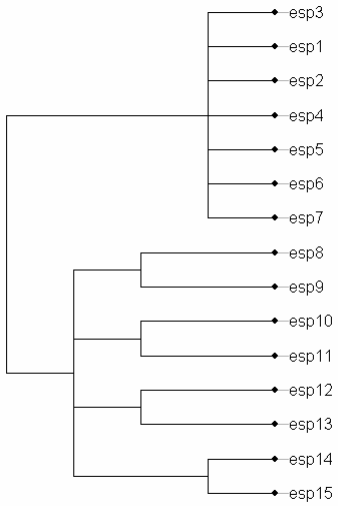
- taxo.eg[[1]] est une taxonomie artificielle à 3 niveaux et 15 taxa
- taxo.eg[[2]] est une taxonomie à deux niveaux et 40 espèces d'oiseaux

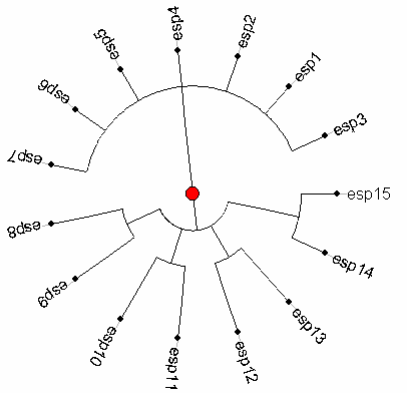
sources

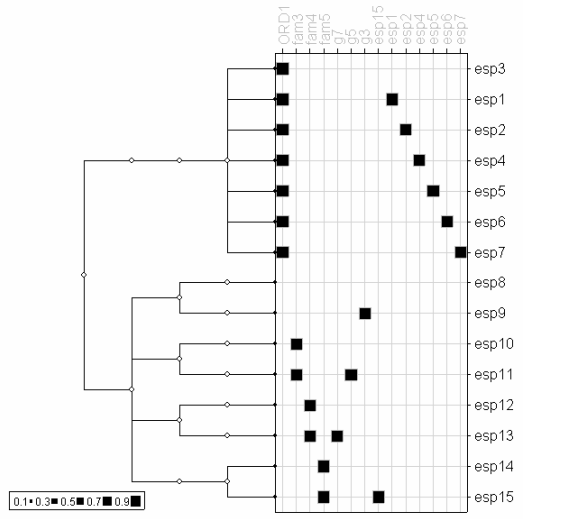
Dolédec, S., Chessel, D., Ter Braak, C.J.F., & Champely, S. (1996) Matching species traits to environmental variables: a new three-table ordination method. Environmental and Ecological Statistics, 3, 143-166.

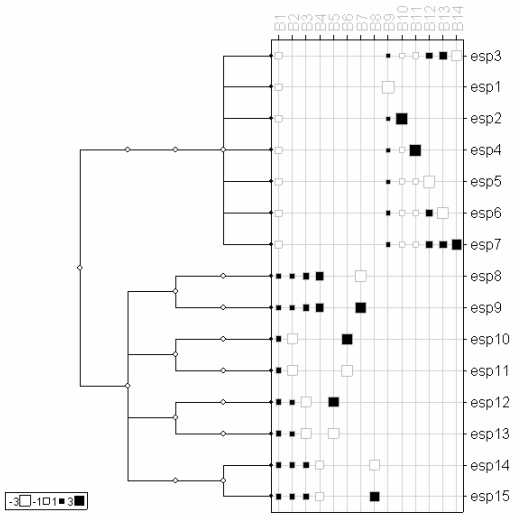
exemples

```
tax <- taxo.eg[[1]]
class(tax)
[1] "data.frame"
tax <- as.taxo(taxo.eg[[1]])
class(tax)
[1] "data.frame" "taxo"
tax
      genre famille ordre
esp8      g2     fam2  ORD2
esp3      g1     fam1  ORD1
esp1      g1     fam1  ORD1
esp2      g1     fam1  ORD1
esp4      g1     fam1  ORD1
esp14     g8     fam5  ORD2
esp15     g8     fam5  ORD2
esp9      g3     fam2  ORD2
esp13     g7     fam4  ORD2
esp12     g6     fam4  ORD2
esp11     g5     fam3  ORD2
esp10     g4     fam3  ORD2
esp5      g1     fam1  ORD1
esp6      g1     fam1  ORD1
esp7      g1     fam1  ORD1

tax.phy <- taxo2phylog(tax)
plot.phylog(tax.phy,
 clabel.leaves = 1)
radial.phylog(tax.phy)

table.phylog(tax.phy$Bindica,tax.phy)
table.phylog(tax.phy$Bscores,tax.phy)
```









23. Variabilité spatiale du taux de chlorophylle dans l'étang de Thau

alias

thau

description

'thau' donne le taux de chlorophylle a (indicateur de la quantité de phytoplancton présente dans l'eau) mesuré pour 63 sites de l'étang de Thau sur une grille régulière dont la maille est de 1 kilomètre.

format

'thau' est une liste à deux composantes :

- *xy* : est un data frame donnant les coordonnées des 63 sites
- *cla* : est un vecteur à 63 élements donnant le taux de chlorophylle a

sources

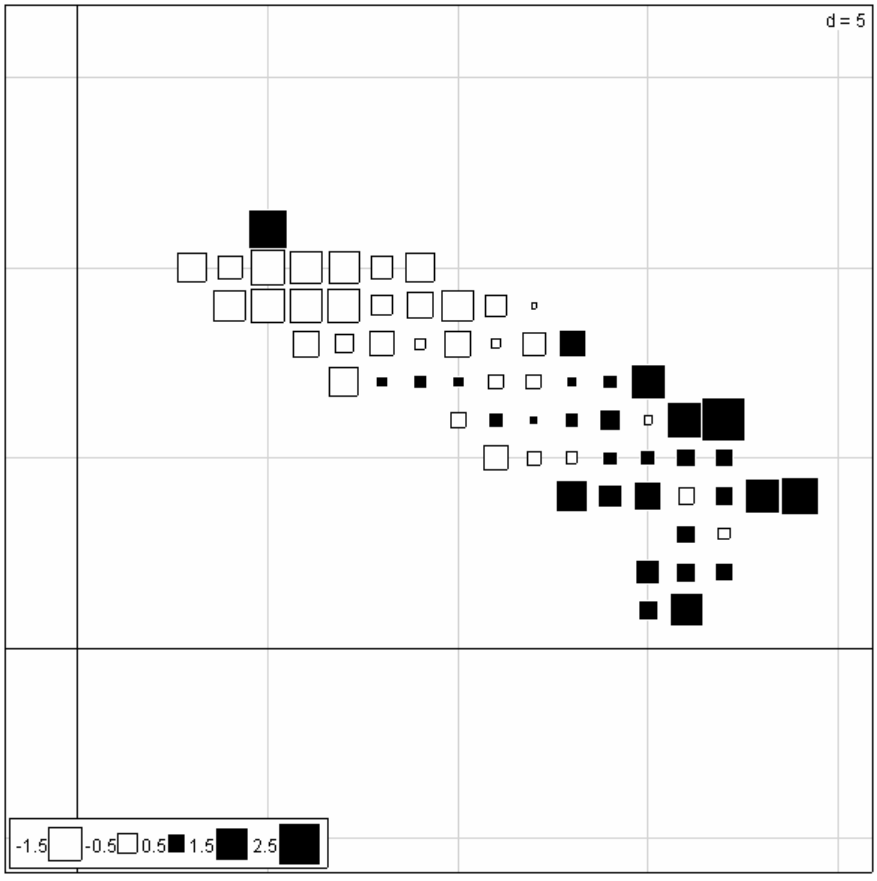
Borcard, D., Legendre, P., Avois-Jacquet, C., & Tuomisto, H. (2004) Dissecting the spatial structure of ecological data at multiple scales. Ecology, 85, 1826-1832.

références

Legendre, P. & Borcard, D. (2003). Quelles sont les échelles spatiales importantes dans un écosystème? In Analyse statistique de données spatiales (eds J.-J. Droesbeke, M. Lejeune & G. Saporta). Technip, Paris.

exemples

```
s.value(thau$xy, scale(thau$cla), csi = 0.75)
```



24. Traits et phylogénie d'ongulés

alias

ungulates

description

'ungulates' donne 4 traits biologiques de 18 espèces d'ongulés ainsi qu'un arbre phylogénétique des espèces.

format

'ungulates' est une liste à deux composantes :

- *tre* : est un arbre phylogénétique au format newick (18 espèces)
- *tab* : est un data frame à 18 lignes (espèces) et 4 colonnes (traits biologiques) :

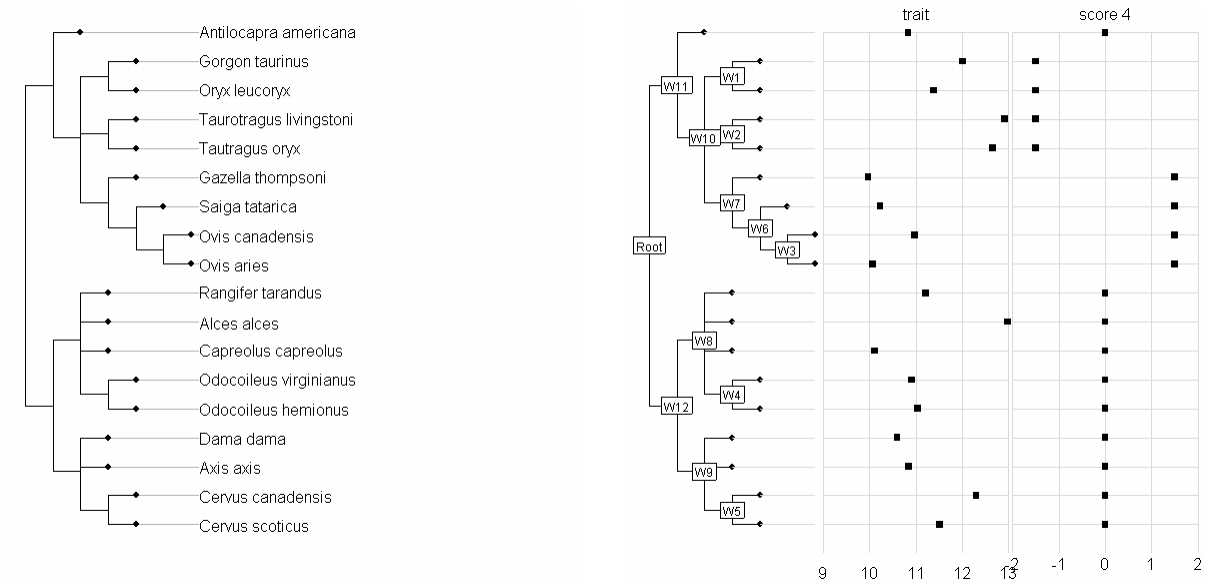
1. " afbw " poids du corps de la femelle adulte (g)
2. " mnw " poids du mâle à la naissance (g)
3. " fnw " poids de la femelle à la naissance (g)
4. " ls " taille de la portée

sources

Pélabon, C., Gaillard, J.M., Loison, A., & Portier, C. (1995) Is sex-biased maternal care limited by total maternal expenditure in polygynous ungulates? Behavioral Ecology and Sociobiology, 37, 311-319.

exemples

```
ung.phy <- newick2phylog(ungulates$tre)
afbw <- log(ungulates$tab$afbw)
plot(ung.phy, f = 0.3)
tab <- cbind.data.frame(afbw, ung.phy$Bscores[,4])
names(tab) <- c("trait", "score 4")
dotchart.phylog(ung.phy,tab, clabel.n = 0.75,
 f = 0.3, scaling = F, ranging = F, joining = F)
```



25. Ordination multiéchelle

alias

vh

description

'vh' est un jeu de données simulé afin d'illustrer les problèmes associés à l'ordination multiéchelle.

format

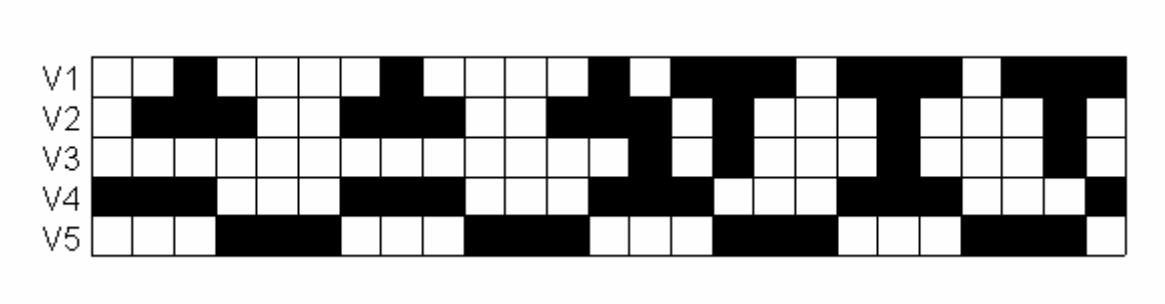
'vh' donne l'absence-presence de 5 espèces fictives sur un transect constitué par 25 placettes.

sources

Ver Hoef, J.M. & Glenn-Lewin, C.G. (1989) Multiscale ordination: a method for detecting pattern at several scales. Vegetatio, 82, 59-67.

exemples

```
table.paint(t(sqrt(vh)), cleg = 0)
```



LES FONCTIONS

1. Représentation graphique d'une ou plusieurs variables le long d'un transect

alias

dotchart.line

description

'dotchart.line' permet la représentation d'une ou plusieurs variables mesurées le long d'un transect. Les mesures peuvent être régulièrement ou irrégulièrement espacées.

usage

```
dotchart.line(values, x.time = 1:nrow(values), scaling = TRUE, ranging = TRUE, yranging = NULL, joining = TRUE, yjoining = NULL, cdot = 0.75, axe1 = TRUE, axe2 = TRUE, lwd.axes = 1.5, csub = 1.5, ...)
```

arguments

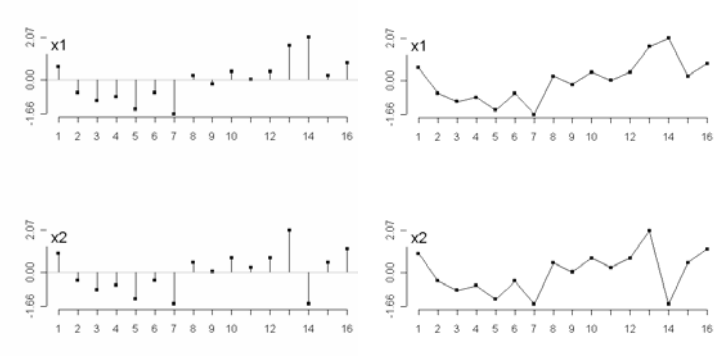
- *values* : est un data frame dont les colonnes correspondent aux variables que l'on souhaite représenter
- *x.time* : est un vecteur dont les éléments correspondent aux coordonnées des points de mesure. Par défaut, les points de mesures sont régulièrement espacés
- *scaling* : est une variable binaire. Si elle prend la valeur TRUE, les variables sont normalisées avant leur représentation.
- *ranging* : est une variable binaire. Si elle prend la valeur TRUE, on adopte une échelle commune pour la représentation de toutes les variables. Par défaut, l'échelle commune est choisie en prenant les valeurs extrêmes sur l'ensemble des variables.
- *yranging* : est un vecteur à deux composantes donnant l'échelle commune à l'ensemble des variables lorsque 'ranging' prend la valeur TRUE
- *joining* : est une variable binaire. Si elle prend la valeur TRUE, les variables sont représentées autour d'un axe horizontal dont la position est donnée par le paramètre 'yjoining'.
- *yjoining* : est un entier donnant la position de l'axe horizontal lorsque 'joining' prend la valeur TRUE. Par défaut, on prend la valeur moyenne de chaque variable.
- *cdot* : est un entier correspondant à la taille des symboles.
- *axe1* : est une variable binaire. Si elle prend la valeur TRUE, l'axe des abscisses est représenté.
- *axe2* : est une variable binaire. Si elle prend la valeur TRUE, l'axe des ordonnées est représenté.
- *lwd*.axes : est un entier correspondant à l'épaisseur des axes
- *csub* : est un entier correspondant à la taille de la légende de chaque représentation. Les légendes sont données par le nom des colonnes du data frame.
- … : arguments supplémentaires

voir également

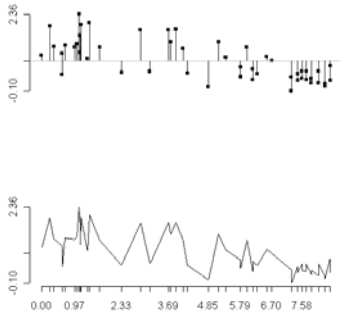
dotchart.phylog

exemples

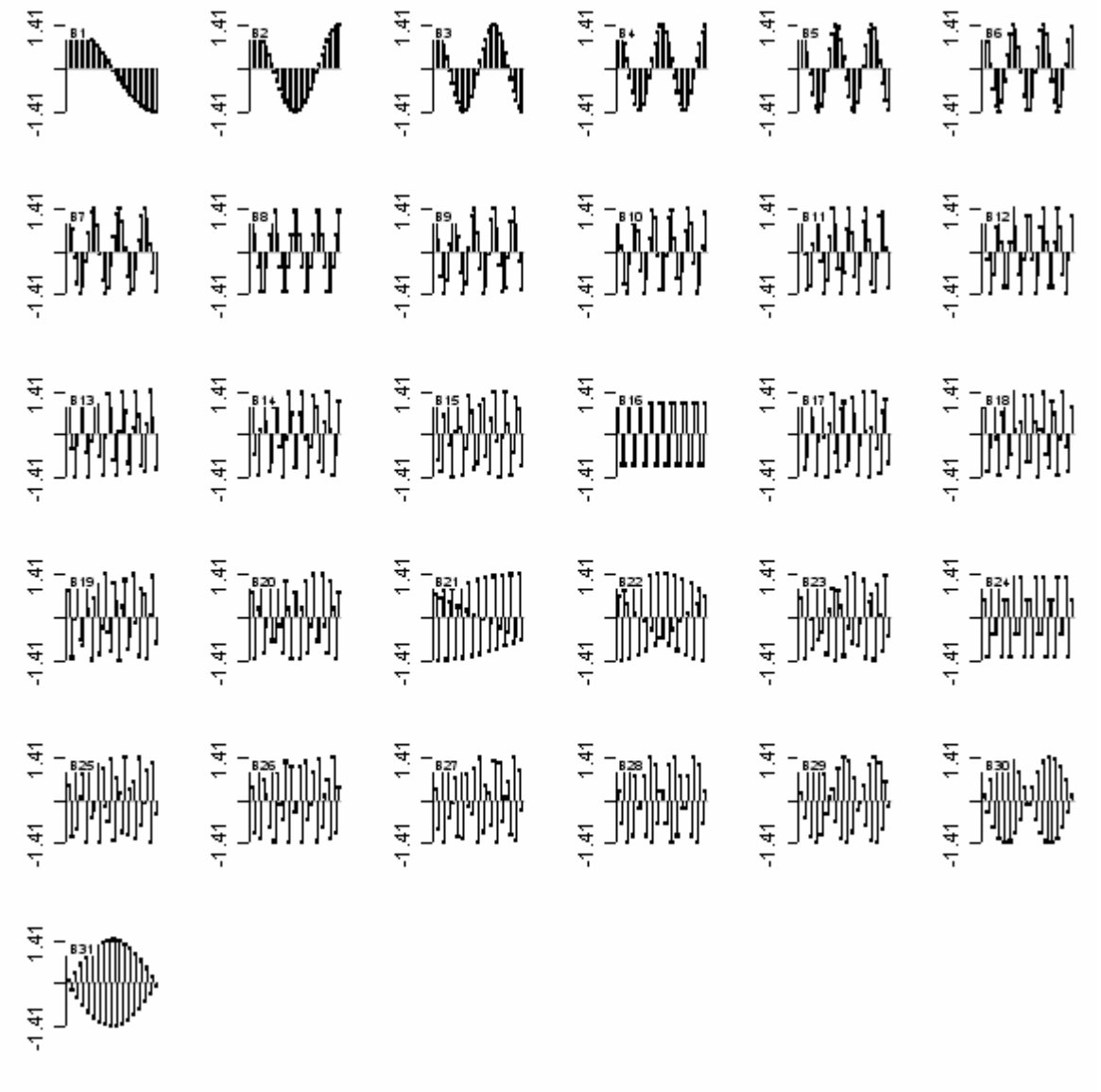
```
x1 <- c(0.2,-0.4,-0.6,-0.5,-0.8,-0.4,
 -0.9,0.0,-0.2,0.1,-0.1,0.1,0.7,0.9,
 0.0,0.3)
x2 <- x1
x2[14] <- - x1[14]
val <- cbind.data.frame(x1, x2)
dotchart.line(val)
dotchart.line(val, joining = FALSE)
```



```
par(mfrow = c(2,1))
dotchart.line(guadeloupe$zooplankton,
 guadeloupe$coord, scaling = FALSE,
 axe1 = FALSE, csub = 0)
dotchart.line(guadeloupe$zooplankton,
 guadeloupe$coord, scaling = FALSE,
 joining = FALSE, cdot = 0, csub = 0)
```




```
orthobas <- orthobasis.line(32)
par(mar = c(3,3,1,1))
dotchart.line(orthobas, csub = 1,
 axe1 = FALSE, cdot = 0.5)
```



R code

```
dotchart.line <- function(values, x.time = 1:nrow(values), scaling = TRUE, ranging
= TRUE, yranging = NULL, joining = TRUE, yjoining = NULL, cdot = 0.75, axe1 = TRUE,
axe2 = TRUE, lwd.axes = 1.5, csub = 1.5, ...){

    if (is.vector(values))
        values <- as.data.frame(values)

    if (!is.data.frame(values))
        stop("'values' is not a data frame")

    if (!is.numeric(as.matrix(values)))
        stop("'values' is not numeric")

n <- nrow(values)
nvar <- ncol(values)
if (nvar > 1)
par(mfrow = n2mfrow(nvar))
names.var <- names(values)

if (scaling == TRUE){
    values <- scalewt(values)
    names(values) <- names.var
    }

for(i in 1:nvar){
m <- mean(values[,i])
    if (ranging == TRUE){
        if (is.null(yranging))  r <- range(values)
            else    r <- yranging
        plot(x.time, values[,i], axes = FALSE, xlab = "", ylab = "", ylim = r, type
= "p", pch = 15, cex = cdot, ...)
        }
        else
            plot(x.time, values[,i], axes = FALSE, xlab = "", ylab = "", type =
"p", pch = 15, cex = cdot, ...)


    if (joining == TRUE){
        if (!is.null(yjoining))  m <- yjoining
        abline(h = m, lwd = 1.5, col = "gray")
        segments(x.time, rep(m, n), x.time, values[,i])
        }
        else
            lines(x.time, values[,i])


    if (axe1 == TRUE)
        axis(1, lwd = lwd.axes, at = x.time)
    if (axe2 == TRUE){
     if (ranging == TRUE)
             axis(2, lwd = lwd.axes, at = round(c(r[1], m, r[2]), 2))
            else
             axis(2,  lwd  =  lwd.axes,  at  =  round(c(range(values[,i])[1],  m,
range(values[,i])[2]), 2))
             }

    scatterutil.sub(names(values)[i], csub = csub, possub = "topleft")
    }
}
```

2. Représentation d'un ou plusieurs traits dans un arbre phylogénétique

alias

dotchart.phylog

description

'dotchart.phylog' représente la phylogénie associée à un ou plusieurs dotplot des traits. Elle est inspirée des travaux de Cleveland (*The dotplot is a graphical method for measurements that have labels*).

usage

```
dotchart.phylog(phylog, values, y = NULL, scaling = TRUE, ranging = TRUE,
yranging = NULL, joining = TRUE, yjoining = NULL, ceti = 1, cdot = 1, csub = 1,
f.phylog = 1/(1 + ncol(values)), ...)
```

arguments

- *phylog* : est un objet de la classe 'phylog'
- *values* : est un data frame dont les colonnes correspondent aux traits que l'on souhaite représenter
- *y* : est un vecteur donnant la position des feuilles pour une représentation donnée de la phylogénie (voir, la fonction enum.phylog)
- *scaling* : est une variable binaire. Si elle prend la valeur TRUE, les variables sont normalisées avant leur représentation.
- *ranging* : est une variable binaire. Si elle prend la valeur TRUE, on adopte une échelle commune pour la représentation de toutes les variables. Par défaut, l'échelle commune est choisie en prenant les valeurs extrêmes sur l'ensemble des variables.
- *yranging* : est un vecteur à deux composantes donnant l'échelle commune à l'ensemble des variables lorsque 'ranging' prend la valeur TRUE
- *joining* : est une variable binaire. Si elle prend la valeur TRUE, les variables sont représentées autour d'un axe horizontal dont la position est donnée par le paramètre 'yjoining'.
- *yjoining* : est un entier donnant la position de l'axe horizontal lorsque 'joining' prend la valeur TRUE. Par défaut, on prend la valeur moyenne de chaque variable.
- *ceti* : est un entier donnant la taille des étiquettes pour la légende
- *cdot* : est un entier donnant la taille des symbols utilisés pour marquer la position des vakeur sur la grille
- *csub* : est une entier donnant la taille des caractères pour la chaîne sub (utilisé avec par(cex)*csub)
- *f.phylog* : est un entier donnant la proportion de la fenêtre consacrée à l'arbre, utile pour laisser la place demandée par les noms des feuilles
- … : arguments supplémentaires

références

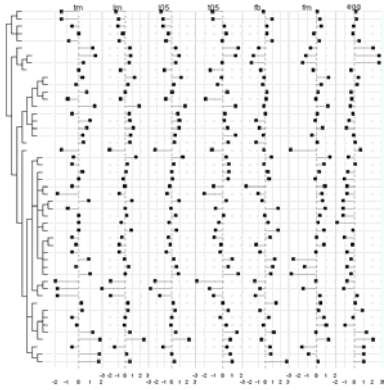
Cleveland, W.S. (1994) The elements of graphing data AT&T Bell Laboratories, Murray Hill, New Jersey.

voir également

symbols.phylog
table.phylog

exemples

```
mjrochet.phy <- newick2phylog(
 mjrochet$tre)
tab0 <- data.frame(
 scalewt(log(mjrochet$tab)))
dotchart.phylog(mjrochet.phy,
 tab0, ceti = 0.5, cdot = 0.75,
 csub = 0.75, cleaves = 0,
 ranging = FALSE)
```



R code

```
"dotchart.phylog" <- function(phylog, values, y = NULL, scaling = TRUE, ranging =
TRUE, yranging = NULL,
 joining = TRUE, yjoining = NULL, ceti = 1, cdot = 1, csub = 1, f.phylog = 1/(1 +
ncol(values)), ...)
{
    if (!inherits(phylog, "phylog"))
        stop("Non convenient data")

    if (is.vector(values))
        values <- as.data.frame(values)

    if (!is.data.frame(values))
        stop("'values' is not a data frame")

    if (!is.numeric(as.matrix(values)))
        stop("'values' is not numeric")

    n <- nrow(values)
    nvar <- ncol(values)
    names.var <- names(values)

    if (length(phylog$leaves) != n)
        stop("Non convenient length")

    if (scaling == TRUE){
        values <- scalewt(values)
        values <- as.data.frame(values)
        names(values) <- names.var
        }

    w <- plot.phylog(x = phylog, y = y, clabel.leaves = 0, f.phylog = f.phylog,
...)
    mar.old <- par("mar")
    on.exit(par(mar = mar.old))
    par(mar = c(0.1, 0.1, 0.1, 0.1))
    par(usr = c(0, 1, -0.05, 1))

    x1 <- w$xbase
    space <- (1 - w$xbase - (w$xbase - max(w$xy$x))/2*nvar)/nvar
    x2 <- x1 + space
    fun1 <- function(x) {x1 + (x2 - x1) * (x - x1.use)/(x2.use - x1.use)}

    for(i in 1:nvar){

    if (ranging == TRUE){
            if (is.null(yranging))
                val.ref <- pretty(range(values), 4)
                else
                    val.ref <- pretty(yranging, 4)
       }
        else
            val.ref <- pretty(values[,i], 4)

    x1.use <- min(val.ref)
    x2.use <- max(val.ref)
    xleg <- fun1(val.ref)
    miny <- 0
    maxy <- max(w$xy$y)
    nleg <- length(xleg)

    segments(xleg, rep(miny, nleg), xleg, rep(maxy, nleg), col = grey(0.85))
    segments(w$xy$x, w$xy$y, rep(max(w$xy$x), n), w$xy$y, col = grey(0.85))
    segments(rep(xleg[1], n), w$xy$y, rep(max(xleg), n), w$xy$y, col = grey(0.85))

    if (cdot > 0)
        points(fun1(values[,i]), w$xy$y, pch = 15, cex = cdot, bg = 1)
```

```
    if (ceti > 0){
        if (trunc(i/2) < (i/2))
            text(xleg, rep((miny - 0.05)*2/3, nleg), as.character(val.ref), cex =
par("cex") * ceti)
            else
                text(xleg, rep((miny - 0.05)*1/3, nleg), as.character(val.ref), cex
= par("cex") * ceti)
        }

    if (joining == TRUE){
        if (is.null(yjoining)) origin <- mean(values[,i])
            else origin <- 0
        segments(fun1(origin), miny, fun1(origin), maxy, lty = 2, col = grey(0.50))
        segments(fun1(values[,i]), w$xy$y, fun1(origin), w$xy$y, col = grey(0.50))
        }

    if (csub > 0)
        text(xleg[3], 1 - (1-max(w$xy$y))/3, names(values)[i], cex = par("cex") *
csub)

    x1 <- x1 + space + (w$xbase - max(w$xy$x))/2
    x2 <- x2 + space + (w$xbase - max(w$xy$x))/2
    }
}
```

3. Test univarié de Geary et Moran

alias

gearymoran

description

'gearymoran' est une fonction mettant en œuvre un test de randomisation basé sur l'indice de Moran. Il est construit afin que le test basé sur l'indice de Moran ou celui basé sur l'indice de Geary soit aussi puissant. En effet, en respectant certaines contraintes telles que celles décrites dans Thioulouse et al. (1997), on a strictement équivalence entre les deux approches.

usage

```
gearymoran(bilis, X, nrepet = 999)
```

arguments

- *bilis* : est une matrice de pondérations de voisinage
- *X* : est un data frame dont les colonnes représentent respectivement les variables dont on veut tester l'absence de structure spatiale
- *nrepet* : est un entier correspondant au nombre de permutations

détails

Soit $\mathbf{L}$ la matrice de pondération de voisinage. On s'assure que la matrice est symétrique par la transformation préalable : $\mathbf{L} = \frac{1}{2}\left(\mathbf{L}' + \mathbf{L}\right)$. On transforme ensuite la matrice $\mathbf{L}$ en matrice de fréquences bivariées. Le terme général de la matrice $\mathbf{L}$ devient donc $l_{ij} = p_{ij} - p_{i.}p_{.j}$ avec $\mathbf{P} = \frac{\mathbf{L}}{\mathbf{1}'\mathbf{L}\mathbf{1}}$. Les termes $p_{i.} = p_{.j}$ représentent la pondération marginale. On centre et l'on normalise chaque variable de $\mathbf{X}$ selon cette pondération. Le test de randomisation est construit sur la statistique de Moran $\mathbf{x}_C'\mathbf{L}\mathbf{x}_C$.

valeurs

'gearymoran' retourne un objet de la classe 'krandtest'

références

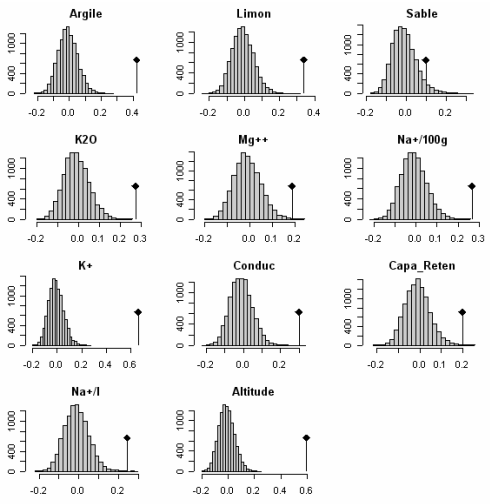
Cliff, A.D. & Ord, J.K. (1973) Spatial autocorrelation Pion, London.
Thioulouse, J., Chessel, D., & Champely, S. (1995) Multivariate analysis of spatial patterns: a unified approach to local and global structures. Environmental and Ecological Statistics, 2, 1-14.

voir également

moran.test, geary.test de la librairie **spdep** pour une version plus classique de ces deux tests

exemples

```
tab0 <- (as.data.frame(scalewt(mafragh$mil)))
bilis0 <- neig2mat(mafragh$neig)
gm0 <- gearymoran(bilis0, tab0, 9999)
gm0
class: krandtest
test number:   11
permutation number:   9999
   test        obs    P(X<=obs) P(X>=obs)
1  Argile      0.4244 1         1e-04
2  Limon       0.338  1         1e-04
3  Sable       0.0995 0.962     0.0382
4  K2O         0.2728 1         2e-04
5  Mg++        0.1858 0.998     0.0022
6  Na+/100g    0.2667 0.9999    3e-04
7  K+          0.6611 1         1e-04
8  Conduc      0.2997 1         1e-04
9  Capa_Reten 0.201  0.9993    9e-04
10 Na+/l       0.243  1         1e-04
11 Altitude    0.5953 1         1e-04

plot(gm0, nclass = 30)
```



R code

```
"gearymoran" <- function (bilis, X, nrepet=999) {
    # bilis doit être une matrice
```

```
    bilis <- as.matrix(bilis)
    nobs <- ncol(bilis)
    # bilis doit être carrée
    if (nrow(bilis) != nobs) stop ("'bilis' is not squared")
    # bilis doit être symétrique
    bilis <- (bilis + t(bilis))/2
    # bilis doit être à termes positifs (voisinages)
    if (any(bilis<0)) stop ("term <0 found in 'bilis'")
    test.names <- names(X)
    X <- data.matrix(X)
    if (nrow(X) != nobs) stop ("non convenient dimension")
    nvar <- ncol(X)
    res <- .C("gearymoran",
        param = as.integer(c(nobs,nvar,nrepet)),
        data = as.double(X),
        bilis = as.double(bilis),
        obs = double(nvar),
        result = double (nrepet*nvar),
        obstot = double(1),
        restot = double (nrepet),
        PACKAGE="ade4"
    )
    res <- c(res$obs,res$result)
    res <- matrix(res, ncol=nvar, byr=TRUE)
    res <- as.data.frame(res)
    names(res) <- test.names
    res <- as.list(res)
    class(res) <- "krandtest"
    return(res)
}

####   ####   ####   ####   Fonction C
####   ####   ####   ####

void gearymoran (int *param, double *data, double *bilis,
    double *obs, double *result, double *obstot, double *restot)
{
    /* Declarations des variables C locales */
    int nobs, nvar, nrepet, i, j, k, krepet, kvar ;
    int *numero;
    double provi;
    double *poili;
    double **mat, **tab, **tabperm;


    /* Allocation memoire pour les variables C locales */
    nobs = param[0];
    nvar = param [1];
    nrepet = param [2];
    vecalloc(&poili,nobs);
    taballoc(&mat,nobs,nobs);
    taballoc(&tab,nobs,nvar);
    taballoc(&tabperm,nobs,nvar);
    vecintalloc (&numero, nobs);

    /* Définitions des variables C locales */
    k = 0;
    for (i=1; i<=nvar; i++) {
        for (j=1; j<=nobs; j++) {
            tab[j][i] = data[k] ;
            k = k+1 ;
       }
    }

    k = 0;
    provi = 0;
    for (j=1; j<=nobs; j++) {
        for (i=1; i<=nobs; i++) {
```

```
            mat[i][j] = bilis[k] ;
            provi = provi +  bilis[k];
            k = k+1 ;
        }
    }
    for (j=1; j<=nobs; j++) {
        for (i=1; i<=nobs; i++) {
            mat[i][j] = mat[i][j]/provi ;
        }
    }
    /* mat contient une distribution de fréquence bivariée */
    for (j=1; j<=nobs; j++) {
        provi = 0;
        for (i=1; i<=nobs; i++) {
            provi = provi + mat[i][j] ;
        }
        poili[j] = provi;
    }
    /* poili contient la distribution marginale
    le test sera du type xtPx avec x centré normé pour la pondération
    marginale et A = QtFQ soit la matrice des pij-pi.p.j */
    matmodifcn(tab,poili);
    /* le tableau est normalisé pour la pondération marginale de la forme*/
    for (j=1; j<=nobs; j++) {
        for (i=1; i<=nobs; i++) {
            mat[i][j] = mat[i][j] -poili[i]*poili[j] ;
        }
    }
    for (kvar=1; kvar<=nvar; kvar++) {
        provi = 0;
        for (j=1; j<=nobs; j++) {
            for (i=1; i<=nobs; i++) {
                provi = provi + tab[i][kvar]*tab[j][kvar]*mat[i][j] ;
            }
        }
        obs[kvar-1] = provi;
    }
    k=0;
    /* les résultats se suivent par simulation */
    for (krepet=1; krepet<=nrepet; krepet++) {
        getpermutation (numero, krepet);
        matpermut (tab, numero, tabperm);
        matmodifcn (tabperm,poili);
        for (kvar=1; kvar<=nvar; kvar++) {
            provi = 0;
            for (j=1; j<=nobs; j++) {
                for (i=1; i<=nobs; i++) {
                    provi = provi + tabperm[i][kvar]*tabperm[j][kvar]*mat[i][j] ;
                }
            }
            result[k] = provi;
            k = k+1;
        }
    }

    /* libération mémoire locale */
    freevec(poili);
    freetab(mat);
    freeintvec(numero);
    freetab(tab);
    freetab(tabperm);
}
```

4. Graphe de type grille complète

alias

gridrowcol

description

'gridrowcol' définit les objets nécessaires à l'analyse de données associées à un graphe de type grille complète pour une relation de la tour à l'ordre 1.

usage

```
gridrowcol(nr , nc, cell.names = NULL)
```

arguments

- *nr* : est un entier correspondant au nombre de lignes de la grille
- *nc* : est un entier correspondant au nombre de colonnes de la grille
- *cell.names* : est un vecteur de chaînes de caractères correspondant aux noms des cellules de la grille

valeurs

- *xy* : est un data frame dont les colonnes donnent les coordonnées du centre des cellules de la grille
- *area* : est un data frame dont les colonnes donnent les coordonnées des sommets des cellules de la grille
- *neig* : est un objet de la classe 'neig' correspondant au graphe de type grille complète pour une relation de la tour à l'ordre 1
- *orthobasis* : est un objet de la classe 'orthobasis' correspondant aux valeurs propres et aux vecteurs propres de l'opérateur de lissage associé au graphe et introduit par Méot

références

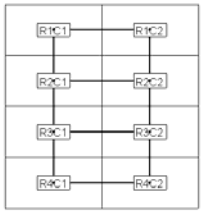
Cornillon, P.-A. (1998) Prise en compte de proximités en analyse factorielle et comparative. Thèse, Ecole Nationale Supérieure Agronomique, Montpellier.
Méot, A., Chessel, D., & Sabatier, D. (1993). Opérateurs de voisnage et analyse des données spatio-temporelles. In Biométrie et environnement (eds J.D. Lebreton & B. Asselain). Masson.

voir également

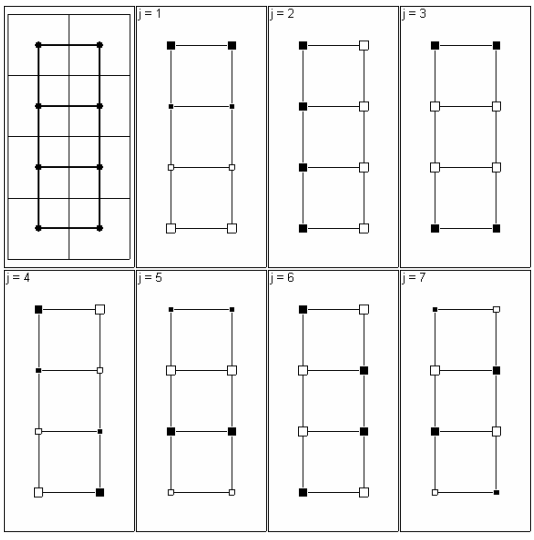
orthobasis

exemples

```
grille <- gridrowcol(4,2)
plot(grille$xy, xlim = c(0,3),
 ylim = c(0,5), type = "n",
  axes = F, xlab = "", ylab = "")
s.label(grille$xy, neig = grille$neig,
 area = grille$area, grid = F, addaxes = F,
 cneig = 2, inc = F, cpoi = 1, add.plot = T)
```




```
# vecteurs propres de l'opérateur de voisinage
par(mfrow = c(2,4))
par(mar = c(0,0,0,0))
plot(grille$xy, xlim = c(0.5,2.5),
 ylim = c(0.5,4.5), type = "n",
 axes = F, xlab = "", ylab   = "")
s.label(grille$xy, neig = grille$neig,
 area = grille$area, grid = F,
 addaxes = F, cneig = 2, inc = F,
 cpoint = 4, clab = 0, add.plot = T)

for(i in 1 :7){
plot(grille$xy, xlim = c(0.5,2.5),
 ylim = c(0.5,4.5), type = "n",
 axes = F, xlab = "", ylab   = "")
s.value(grille$xy, grille$orthobas[,i],
 neig = grille$neig, grid = F,
 addaxes = F, cneig = 0.5,
 inc = F, add.plot = T, cleg = 0,
 sub = paste("j =", i, sep = " "),
 csub = 2)}
```



R code

```
"gridrowcol" <- function (nr , nc, cell.names = NULL) {
    nr <- as.integer(nr)
    if (nr < 1) stop("nr nonpositive")
    nc <- as.integer(nc)
    if (nc < 1)  stop("nc nonpositive")
    ncell <- nr*nc

    # xy : coordonnées des centres des cadrats de la grille
    xy <- matrix(0,nr,nc)
    xy <- cbind(as.numeric(t(col(xy))),as.numeric(t(row(xy))))
    if (!is.null(cell.names)) {
        if (length(cell.names) != nr*nc) names <- NULL
    }
    if (is.null (cell.names)) {
        cell.names <- paste("R",xy[,2],"C",xy[,1],sep="")
    }
    xy <- data.frame(xy)
    names(xy)=c("x","y")
    row.names(xy) <- cell.names
    xy$"y" <- nr + 1 - xy$"y"
    res <- list(xy = xy)

    # area : cadrats de la grille
    area <- rep(row.names(xy), rep(4,ncell))
    area <- as.factor(area)
    w <- cbind(xy$"x" - 0.5, xy$"x" - 0.5, xy$"x" + 0.5, xy$"x" + 0.5)
    w <- as.numeric(t(w))
    area <- cbind.data.frame(area,w)
    w <- cbind(xy$"y"-0.5,xy$"y"+0.5,xy$"y"+0.5,xy$"y"-0.5)
    w <- as.numeric(t(w))
    area <- cbind.data.frame(area,w)
    names(area) <- c("cell","x","y")
    res$area <- area

    # neig : graphe de voisinage avec relation de la tour
    d0 <- dist2mat(dist.quant(xy,1))
    d0 <- 1*(d0 < 1.2)
    diag(d0) <- 0
    pvoisi <- unlist(apply(d0,1,sum))
    naret <- sum(pvoisi)
    pvoisi <- pvoisi/naret
    d0 <- neig(mat01 = d0)
    res$neig <- d0

    # orthobasis: base orthonormée
    xy$"y" <- nr + 1 - xy$"y"
    "glin" <- function (n) {
        n <- n
        "vecpro" <- function(k) {
            x <- cos(k*pi*((1:n)-0.5)/n)
            x <- x/sqrt(sum(x*x))
            # print(x)
        }
        w <- unlist(lapply(0:(n-1),vecpro))
        w <- matrix(w,n)
    }

    orthobasis <- glin(nr)%x%glin(nc)
    pirow <- pi/nr
    picol<- pi/nc
    salpha <- (sin((0:(nr-1))*pirow/2))^2
    sbeta <- (sin((0:(nc-1))*picol/2))^2
    z <- rep(sbeta,nr)+rep(salpha,rep(nc,nr))
    z <- 4*z/nr/nc
    w <- order(z)[-1]
    z <- z[w]
    orthobasis <- sqrt(ncell)*orthobasis[,w]
```

```
    orthobasis <- data.frame(orthobasis)
    val <- unlist(lapply(orthobasis,function(x) sum(x*x*pvoisi)))
    val <- val - z*ncell*ncell/naret
    ord <- rev(order(val))
    orthobasis <- orthobasis[,ord]
    val <- val[ord]
    names(orthobasis) = paste("S",1:(ncell-1),sep="")
    row.names(orthobasis) = row.names(res$xy)
    attr(orthobasis,"values") <- val
    attr(orthobasis,"weights") <- rep(1/ncell,ncell)
    attr(orthobasis,"call") <- match.call()
    attr(orthobasis,"class") <- c("orthobasis","data.frame")
    res$orthobasis <- orthobasis
    return(res)
}
```

5. *k* formes bilinéaires symétriques

alias

knb2kfbs
ttlv.kfbs
msbs.kfbs
orthobasis2kfbs
summary.kfbs
subset.kfbs
cbind.kfbs
as.matrix.kfbs

description

Ces fonctions créent et manipulent des objets de la classe 'kfbs'. Ils définissent des familles de *k* formes bilinéaires symétriques. Une forme bilinéaire symétrique est une matrice carrée symétrique $\mathbf{A}$ à *n* lignes et *n* colonnes. Les *k* formes bilinéaires symétriques sont rangées en colonnes dans une matrice qui contient les demi matrices inférieures [(1,1), (2,1), …, (*n*,1), (2,2), …, (*n*,2), …, (*n*,*n*)].

Les attributs d'un objet de la classe 'kfbs' sont :

- $dim : la dimension de la matrices de stockage (*n**(*n*+1)/2, *k*)
- $npoints : le nombre de points (*n*)
- $nforms : le nombre de formes bilinéaires symétriques (*k*)
- $scalprod : prend la valeur TRUE si les formes sont positives, FALSE sinon
- $trace1 : les traces des matrices $\mathbf{A}_k$
- $trace2 : les traces des matrices $\mathbf{A}_k^2$
- $sum : les sommes des matrices $\mathbf{A}_k$
- $rang : les rangs des matrices $\mathbf{A}_k$
- $norm : les normes spectrales des matrices $\mathbf{A}_k$
- $labels : les noms donnés aux matrices $\mathbf{A}_k$
- $call : le nom de la commande qui a permis la création de l'objet
- $class : la classe de l'objet, à savoir 'kfbs'

'knb2kfbs' définit les formes associées aux graphes de voisinage
'ttlv.kfbs' définit les formes associées aux métriques de Hill ou TTLV
'msbs.kfbs' définit les formes associées aux métriques de Noy-Meir Anderson ou MSBS
'orthobasis.kfbs' définit les formes associées aux projecteurs définit par la partition des vecteurs d'une base orthonormée
'summary.kfbs' donne les attributs d'un objet de la classe 'kfbs'
'subset.kfbs' définit une sous famille de *l* formes bilinéaires symétriques à partir d'un objet de la classe 'kfbs'
'cbind.kfbs' associe deux familles de *l* et *k* formes bilinéaires symétriques en une famille de *l*+*k* formes bilinéaires symétriques
'as.matrix.kfbs' extrait les matrices symétriques correspondant aux formes bilinéaires symétriques

usage

```
knb2kfbs(knb, method = "Moran")
msbs.kfbs (n = 64, tbloc = c(1, 2, 4, 8, 16, 32, 64))
ttlv.kfbs (n = 32, tbloc = c(1, 2, 4, 8, 16), method = "Moran")
orthobasis2kfbs (orthobas, level)
summary.kfbs (kfbs)
subset.kfbs(kfbs, select)
cbind.kfbs(...)
as.matrix.kfbs (kfbs, k = 1)
```

arguments

- *knb* : un objet de la classe 'knb'. Il correspond à *k* graphes de voisinage
- *method* : une chaîne de caractères. Si "Moran" calcule la forme $\mathbf{M}$, sinon la forme positive $\mathbf{N}-\mathbf{M}$
- *n* : un entier correspondant au nombre de points
- *tbloc* : un vecteur d'entier correspondant à la taille des blocs
- *orthobas* : un objet de la classe 'orthobasis'

- *level* : un facteur définissant une partition des vecteurs de la base orthonormée
- *kfbs* : un objet de la classe 'kfbs'
- *select* : un vecteur d'entiers correspondant aux numéros des formes que l'on souhaite conserver
- … : plusieurs objets de la classe 'kfbs'
- *k* : un entier correspondant au numéro de la forme dont on veut extraire la matrice correspondante

details

Les matrices symétriques $\mathbf{A}_k$ à *n* lignes et *n* colonnes de nombres réels définissent les formes bilinéaires symétriques sur $\mathbb{R}^n$:

$$f_{\mathbf{A}_k}(\mathbf{x},\mathbf{y}) = \mathbf{x}^t\mathbf{A}_k\mathbf{y} = \mathbf{y}^t\mathbf{A}_k\mathbf{x}$$

et les formes quadratiques :

$$q_{\mathbf{A}_k}(\mathbf{x}) = \mathbf{x}^t\mathbf{A}_k\mathbf{x} \cdot$$

Les fonctions proposées ont pour but de définir des familles de matrices $\mathbf{A}_k$ à partir d'objets diverses. Un graphe de voisinage donne deux matrices symétriques, celle associée au calcul de l'indice de Moran, notée $\mathbf{M}$ et celle associée à l'indice de Geary, notée $\mathbf{N}-\mathbf{M}$. Les métriques introduites par Hill définissant également deux matrices symétriques, suivant que l'on intègre le voisinage entre blocs sous la forme $\mathbf{M}$ ou $\mathbf{N}-\mathbf{M}$. Les métriques introduites par Greig-Smith, reprises par Noy-Meir, associée à l'analyse de variance hiérarchique, définissent des bases orthonormées $\mathbf{U}_k$ et les matrices symétriques associées aux projecteurs $\mathbf{\Pi}_k = \mathbf{U}_k\mathbf{U}_k^t$. De manière générale, à toute base orthonormée, on peut associer un ensemble de projecteurs $\mathbf{\Pi}_k = \mathbf{U}_k\mathbf{U}_k^t$ avec $\sum_k \mathbf{\Pi}_k = \mathbf{Id}_n$.

valeurs

Ces fonctions retournent des objets de la classe 'kfbs', à l'exception de la fonction 'as.matrix.kfbs' qui retourne une matrice.

références

Banet, T.A. & Lebart, L. (1984). Local and Partial Principal Component Analysis (PCA) and Correspondence Analysis (CA). In COMPSTAT 84 (ed I.A.f.S. Computing.), pp. 113-123. Physica-Verlag, Vienna.

Di Bella, G. & Jona-Lasinio, G. (1996) Including spatial contiguity information in the analysis of multispecific patterns. Environmental and Ecological Statistics, 3, 269-280.

Geary, R.C. (1954) The contiguity ratio and statistical mapping. The incorporated Statistician, 5, 115-145.

Hill, M.O. (1973) The intensity of spatial pattern in plant communities. Journal of Ecology, 61, 225-235.

Moran, P.A.P. (1948) The interpretation of statistical maps. Journal of the Royal Statistical Society, B, 10, 243-251.

Noy-Meir, I. & Anderson, D.J. (1971). Multivariate pattern analysis, or multiscale ordination: towards a vegetation hologram ? In Statistical Ecology, III Many species populations ecosystems and systems analysis (eds G.P. Patil, E.C. Pielou & W.E. Waters), pp. 208-231. Pennsylvania State University Press.

Smouse, P. & Peakall, R. (1999) Spatial autocorrelation analysis of individual multiallele and multilocus genetic structure. Heredity, 82, 561-573.

voir également

val.kfbs
val.kfbs.rtest
statis.kfbs
spectral.kfbs

exemples

```
# graphe linéaire
ng <- neig(n.line = 8)
# puissances du graphes
knb <- neig2knb(ng)

# formes de Geary/Lebart
geary.kfbs <- knb2kfbs(knb, method = "Geary")
# formes de Moran/Smouse
moran.kfbs <- knb2kfbs(knb, method = "Moran")
# formes de Noy-Meir/Anderson
noy.kfbs <- msbs.kfbs(n = 8, tbloc = c(1,2,4,8))
```

```
# formes de Hill
hill.geary.kfbs <- ttlv.kfbs(n = 8, tbloc = c(1,2,4), method = "Geary")
hill.moran.kfbs <- ttlv.kfbs(n = 8, tbloc = c(1,2,4), method = "Moran")
```

R code

```
"knb2kfbs" <- function(knb, method = "Moran") {
    # renvoie les demi-matrices de forme bilinéaires symétriques associées à k
       # graphes de voisinage
    # method = "Moran" les Mk
    # method = "Geary" les Nk-Mk
    # le résultat est un objet de la classe kfbs (demi-tableaux en colonnes)
    if (!inherits(knb, "knb")) stop("object of class 'knb' expected")

nmet <- length(knb)
n <- length(knb[[1]])
res <- matrix(0, n*(n+1)/2, nmet)
trace <- rep(0, nmet); tracecarre <- trace ; s <- trace ; rank <- trace; ev <-
trace

    if (method == "Geary") {
        for (i in 1:nmet) {
            M <- neig2mat(nb2neig(knb[[i]]))
            N <- diag(apply(as.data.frame(M), 1, sum))
            M <- N-M
            res[, i] <- as.vector(M)[col(M) <= row(M)]
            trace[i] <- sum(diag(M))
            tracecarre[i] <- sum(diag(M%*%M))
            s[i] <- sum(M)
            rank[i] <- qr(M)$rank
            ev[i] <- sqrt(eigen(M%*%M)$values[1])
        }
        attr(res, "npoints") <- n
        attr(res, "nforms") <- nmet
        attr(res, "scalprod") <- T
        attr(res, "trace1") <- trace
        attr(res, "trace2") <- tracecarre
        attr(res, "sum") <- s
        attr(res, "rank") <- rank
        attr(res, "norm") <- ev
        attr(res, "labels") <- paste("G", 1:nmet, sep = "")
        attr(res, "call") <- match.call()
        class(res) <- "kfbs";
        return (res)
    }

    if (method == "Moran") {
        for (i in 1:nmet) {
            M <- neig2mat(nb2neig(knb[[i]]))
            res[, i] <- as.vector(M)[col(M) <= row(M)]
            tracecarre[i] <- sum(diag(M%*%M))
            trace[i] <- sum(diag(M))
            s[i] <- sum(M)
            rank[i] <- qr(M)$rank
            ev[i] <- sqrt(eigen(M%*%M)$values[1])
        }
        attr(res, "npoints") <- n
        attr(res, "nforms") <- nmet
        attr(res, "scalprod") <- F
        attr(res, "trace1") <- trace
        attr(res, "trace2") <- tracecarre
        attr(res, "sum") <- s
        attr(res, "rank") <- rank
        attr(res, "norm") <- ev
        attr(res, "labels") <- paste("M", 1:nmet, sep = "")
        attr(res, "call") <- match.call()
        class(res) <- "kfbs";
        return (res)
```

```
    }
    stop("Non convenient method")
}

####   ####   ####   ####
####   ####   ####   ####

"msbs.kfbs" <- function(n = 64, tbloc = c(1, 2, 4, 8, 16, 32, 64)) {
    # renvoie les demi-matrices de forme bilinéaires symétriques associées à une
       # analyse hierarchique de la variance
    # le résultat est un objet de la classe kfbs (demi-tableaux en colonnes)

    disj <- function(a){
        if (!is.factor(a)) stop ("factor expected")
        n <- length(a); m <- nlevels(a)
        adi <- matrix(0, n, m)
        for (i in 1:n) adi[i, a[i]] <- 1
        return (adi)
    }

    ainterbortho <- function(a, b){
        if (!is.factor(a)) stop ("a: factor expected")
        if (!is.factor(b)) stop ("b: factor expected")
        if (length(a) != length(b)) stop ("length non matched")
        A <- disj(a)
        B <- disj(b)
        qrA <- qr(A)
        qrB <- qr(B)
        if (qrA$rank == 0) return(list(dim = 0, base = NULL))
        if (qrB$rank == length(a)) return(list(dim = 0, base = NULL))
        A0 <- qr.Q(qrA, complet = F)[, 1:qrA$rank] #base orthonormée de A
        B0 <- qr.Q(qrB, complet = T)[, ((qrB$rank+1):(length(a)))]
             #base orthonormée de B orthogonale

        C0 <- t(A0)%*%B0
        eig0 <- eigen(t(C0)%*%C0, sym = T)
        dim <- sum(((1-eig0$values)^2) < 1e-07)
        #analyse canonique entre B ortho
        # et B. Si on garde les 0, on a les vecteurs propres de B (théorème de la
        # base incomplète)
        # ces vecteurs là sont-ils les mêmes que ceux de B0?
        base <- B0%*%eig0$vectors[, 1:dim]
        #print(eig0$values)
        return(list(dim = dim, base = base, d = eig0$values))
    }

nmet <- length(tbloc)-1; res <- matrix(0, n*(n+1)/2, nmet)
trace <- rep(0, nmet); tracecarre <- trace; s <- trace; r0 <- trace; ev <- trace
labels <- rep("", nmet)

for (i in 1:nmet) {
    tbloc1 <- tbloc[i]; nbloc1 <- (n%/%tbloc1)+1 #1 de trop si complet
    a <- rep(1:nbloc1, rep(tbloc1, nbloc1))
    a <- as.factor(a[1:n])
    tbloc2 <- tbloc[i+1]; nbloc2 <- (n%/%tbloc2)+1 #1 de trop si complet
    b <- rep(1:nbloc2, rep(tbloc2, nbloc2))
    b <- as.factor(b[1:n])
    w <- ainterbortho(a, b)
    if (w$dim != 0) {
        M <- w$base%*%t(w$base)
        res[, i] <- as.vector(M)[col(M)<=row(M)]
        trace[i] <- sum(diag(M))
        tracecarre[i] <- sum(diag(M%*%M))
        s[i] <- sum(M)
        r0[i] <- qr(M)$rank
        ev[i] <- sqrt(eigen(M%*%M)$values[1])
        labels[i] <- paste(tbloc1, tbloc2, sep = "_")
    }
```

```
}

attr(res, "npoints") <- n
attr(res, "nforms") <- nmet
attr(res, "scalprod") <- T
attr(res, "trace1") <- trace
attr(res, "trace2") <- tracecarre
attr(res, "sum") <- s
attr(res, "rank") <- r0
attr(res, "norm") <- ev
attr(res, "labels") <- labels
class(res) <- "kfbs";
attr(res, "call") <- match.call()
return (res)
}

####   ####   ####   ####
####   ####   ####   ####

"ttlv.kfbs" <- function(n = 32, tbloc = c(1, 2, 4, 8, 16), method = "Moran") {
    # renvoie les demi-matrices de forme bilinéaires symétriques des métriques
       # associées aux tailles de blocs
    # method = "Geary" calcul les métriques N-M
    # method = "Moran" calcul les métriques M
    # le résultat est un objet de la classe kfbs (demi-tableaux en colonnes)

nmet <- length(tbloc) # il y aura nmet métriques
res <- matrix(0, n*(n+1)/2, nmet)
trace1 <- rep(0, nmet) ; trace2 <- rep(0, nmet) ; s <- rep(0, nmet); r0 <- s ; ev
<- s

for (i in nmet:1) {
    taillebloc <- tbloc[i] ; nbloc <- n-taillebloc+1
    a <- matrix(0, n, nbloc)
    for (j in 1:nbloc) a[j:(j+taillebloc-1), j] <- 1
    b <- matrix(0, nbloc, nbloc)
    for (k in 1:(nbloc-taillebloc)){
        b[k, k+taillebloc] <- b[k+taillebloc, k] <- 1
    }
    if (method == "Geary") {
        b <- diag(apply(b, 1, sum))-b
        a <- a%*%b%*%t(a)
    }
    if (method == "Moran") {
        a <- a%*%b%*%t(a)
    }
    res[, i] <- as.vector(a)[col(a)<=row(a)]
    trace1[i] <- sum(diag(a))
    trace2[i] <- sum(diag(a%*%a))
    s[i] <- sum(a)
    r0[i] <- qr(a)$rank
    ev[i] <- sqrt(eigen(a%*%a)$values[1])
}

attr(res, "npoints") <- n
attr(res, "nforms") <- nmet
if (method == "Geary") attr(res, "scalprod") <- TRUE
if (method == "Moran") attr(res, "scalprod") <- FALSE
attr(res, "trace1") <- trace1
attr(res, "trace2") <- trace2
attr(res, "sum") <- s
attr(res, "rank") <- r0
attr(res, "norm") <- ev
attr(res, "call") <- match.call() ;
if (method == "Moran") attr(res, "labels") <- paste ("ttm", tbloc, sep = " <- ")
if (method == "Geary") attr(res, "labels") <- paste ("ttg", tbloc, sep = " <- ")
class(res) <- "kfbs" ;
return (res)
```

```
}

####   ####   ####   ####
####   ####   ####   ####

"orthobasis2kfbs" <- function(orthobas, level){
    # renvoie les demi-matrices de forme bilinéaires symétriques associées à une
       # base orthonormée
    # le résultat est un objet de la classe kfbs (demi-tableaux en colonnes)
    #  il  s'agit  des  projecteurs  associés  aux  groupes  de  vecteurs  propres  :
uk%*%t(uk)
    if  (!inherits(orthobas,  "orthobasis"))  stop("object  of  class  'orthobasis'
expected")
    if (!is.factor(level)) stop("'factor' expected")

n <- nrow(orthobas)
nmet <- nlevels(level)
level.names <- levels(level)
res <- matrix(0, n*(n+1)/2, nmet)
trace <- rep(0, nmet); tracecarre <- trace ; s <- trace

for (i in 1:nmet) {
    u <- (1:(n - 1))[level == level.names[i]]
    a <- as.matrix(orthobas)[, u]; a <- a%*%t(a)
    res[, i] <- as.vector(a)[col(a)<=row(a)]
    trace[i] <- sum(a*a)
    tracecarre[i] <- sum(a*a)
    s[i] <- sum(a)
}

attr(res, "npoints") <- n
attr(res, "nforms") <- nmet
attr(res, "scalprod") <- T
attr(res, "trace1") <- trace
attr(res, "trace2") <- tracecarre
attr(res, "sum") <- s
attr(res, "rank") <- rep(1, nmet)
attr(res, "norm") <- rep(1, nmet)
attr(res, "labels") <- level.names
attr(res, "call") <- match.call() ;
class(res) <- "kfbs" ;
return (res)
}

####   ####   ####   ####
####   ####   ####   ####

"summary.kfbs" <- function(kfbs) {
    if (class(kfbs) != "kfbs") stop ("Non convenient data")

cat ("K bilinear symetric forms\n")
n <- attr(kfbs, "npoints") ; cat("Points : ", n)
p <- attr(kfbs, "nforms") ; cat("    Forms : ", p, "\n")
a <- attr(kfbs, "scalprod") ; cat("All positive forms : ", a, "\n")
cat("Call : "); print(attr(kfbs, "call"))
if (p == 1) {
    w  <-  c(attr(kfbs,  "trace1"),  attr(kfbs,  "trace2"),  attr(kfbs,  "sum"),
attr(kfbs, "rank"), attr(kfbs, "norm"))
    names(w) <- c("tr(f)", "tr(f^2)", "sum", "rank", "norm")
    return(w)
}

w  <-  cbind.data.frame(attr(kfbs,  "trace1"),  attr(kfbs,  "trace2"),  attr(kfbs,
"sum"), attr(kfbs, "rank"), attr(kfbs, "norm"))
row.names(w) <- attr(kfbs, "labels")
names(w) <- c("tr(f)", "tr(f^2)", "sum", "rank", "norm")
return(w)
}
```

```
####   ####   ####   ####
####   ####   ####   ####

"subset.kfbs" <- function(kfbs, select){
    if (class(kfbs)!="kfbs") stop ("Non convenient data")

n <- attr(kfbs, "npoints"); p <- attr(kfbs, "nforms");
nl <- as.list(1:ncol(kfbs))
names(nl) <- names(kfbs)
vars <- eval(substitute(select), nl, sys.frame(sys.parent()))
res <- kfbs[, vars, drop = FALSE]
attr(res, "npoints") <- n
attr(res, "nforms") <- ncol(res)
attr(res, "scalprod") <- attr(kfbs, "scalprod")
attr(res, "trace1") <- attr(kfbs, "trace1")[vars]
attr(res, "trace2") <- attr(kfbs, "trace2")[vars]
attr(res, "sum") <- attr(kfbs, "sum")[vars]
attr(res, "rank") <- attr(kfbs, "rank")[vars]
attr(res, "norm") <- attr(kfbs, "norm")[vars]
attr(res, "labels") <- attr(kfbs, "labels")[vars]
attr(res, "call") <- match.call()
class(res) <- "kfbs"
return (res)
}

####   ####   ####   ####
####   ####   ####   ####

"cbind.kfbs" <- function(...) {
    listobj <- list(...)
    if (any(sapply(listobj, class) != "kfbs")) stop ("Non convenient data")
    n <- mean(sapply(listobj, function(x) attr(x, "npoints")))
    if (any(sapply(listobj, function(x) attr(x, "npoints")) != n))
        stop ("Not all forms have the same dimension")

p <- sum(sapply(listobj, function(x) attr(x, "nforms")))
trace1 <- unlist(sapply(listobj, function(x) attr(x, "trace1")))
trace2 <- unlist(sapply(listobj, function(x) attr(x, "trace2")))
s0 <- unlist(sapply(listobj, function(x) attr(x, "sum")))
r0 <- unlist(sapply(listobj, function(x) attr(x, "rank")))
labels <- unlist(sapply(listobj, function(x) attr(x, "labels")))
scalprod <- all(sapply(listobj, function(x) attr(x, "scalprod")))
ev <- unlist(sapply(listobj, function(x) attr(x, "norm")))
X <- matrix(0, n*(n+1)/2, p)
k <- 0
for (i in 1:(length(listobj))) {
    X1 <- listobj[[i]]
    m <- attr(X1, "nforms")
    X[, ((k+1):(k+m))] <- X1
    k <- k+m
}

attr(X, "npoints") <- n
attr(X, "nforms") <- p
attr(X, "scalprod") <- scalprod
attr(X, "trace1") <- trace1
attr(X, "trace2") <- trace2
attr(X, "sum") <- s0
attr(X, "rank") <- r0
attr(X, "norm") <- r0
attr(X, "labels") <- labels
attr(X, "call") <- match.call() ;
class(X) <- "kfbs"
return (X)
}

####   ####   ####   ####
```

```
####   ####   ####   ####

"as.matrix.kfbs" <- function(kfbs, k = 1) {
    if (class(kfbs)!="kfbs") stop ("Non convenient data")

n <- attr(kfbs, "npoints"); p <- attr(kfbs, "nforms")
a <- matrix(0, n, n)
if (p == 1) w <- kfbs else w <- obj[, k]
a[row(a) >= col(a)] <- w
a <- a + t(a)
a <- a-diag(diag(a)/2)
return (a)
}
```

6. Ordination multi échelles version Geary

alias

kmsov

description

'kmsov' réalise séparément k ordinations sous contrainte de type 'msov'. Lorsque les matrices de contiguïté de chaque analyse constituent une partition de l'ensemble des couples de voisins, on obtient une décomposition de l'inertie du triplet d'origine. C'est le principe de l'ordination multiéchelle défini par Noy-Meir et repris par Ver-Hoef.

usage

```
kmsov(dudi, knb)
```

arguments

- *dudi* : un objet de la classe 'dudi'. Cet objet est le résultat de l'analyse du triplet statistique $(\mathbf{X},\mathbf{Q},\mathbf{D})$
- *knb* : un objet de la classe 'knb'.

détails

On note $(\mathbf{X}_C,\mathbf{Q},\mathbf{D})$ le triplet de l'analyse statistique considérée. On note $\mathbf{L}_h$ la matrice de contiguïté du graphe de voisinage *h*. On introduit les matrices $\mathbf{M}_h = \mathbf{D}\mathbf{L}_h\mathbf{D}$ et $\mathbf{N}_h = diag(\mathbf{E}_h)$ avec $\mathbf{E}_h = \mathbf{M}_h\mathbf{1}_n$. La partition de l'ensemble des couples de voisins se traduit par une décomposition de l'inertie. En effet, on a d'une part $\sum_h(\mathbf{N}_h - \mathbf{M}_h) = \mathbf{N} - \mathbf{M}$ et $\mathbf{Q}^{1/2}\mathbf{X}_C^t\mathbf{D}\mathbf{X}_C\mathbf{Q}^{1/2} = \mathbf{Q}^{1/2}\mathbf{X}_C^t(\mathbf{N}-\mathbf{M})\mathbf{X}_C\mathbf{Q}^{1/2}$ d'où $tr\left(\mathbf{Q}^{1/2}\mathbf{X}_C^t(\mathbf{N}-\mathbf{M})\mathbf{X}_C\mathbf{Q}^{1/2}\right) = \sum_h tr\left(\mathbf{Q}^{1/2}\mathbf{X}_C^t(\mathbf{N}_h-\mathbf{M}_h)\mathbf{X}_C\mathbf{Q}^{1/2}\right)$. L'idée d'origine est dans la réécriture de la variance par Lebart. C'est l'application de la mso au cas particulier d'une variable quantitative dans la logique de l'ACP. La remarque essentielle tient donc dans le fait que la variance peut se réecrire sous la forme $\text{var}(x_{.p}) = \frac{1}{2}\sum_{i=1}^{n}\sum_{i=1}^{n} d_{ii}d_{jj}(x_{ip}-x_{jp})^2 = x_{.p}^t(\mathbf{N}-\mathbf{M})x_{.p}$.

valeurs

'kmsov' retourne un objet de la classe kmsov. Il s'agit d'une liste d'objets de la classe 'msov'.

références

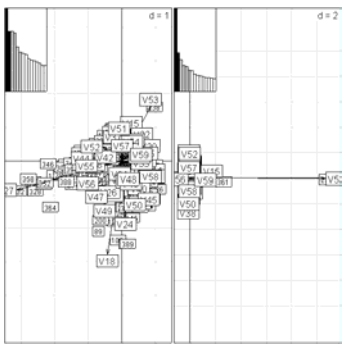
Couteron, P. & Ollier, S. (sous presse) A generalized variogram-based framework for multiscale ordination. Ecology.

voir également

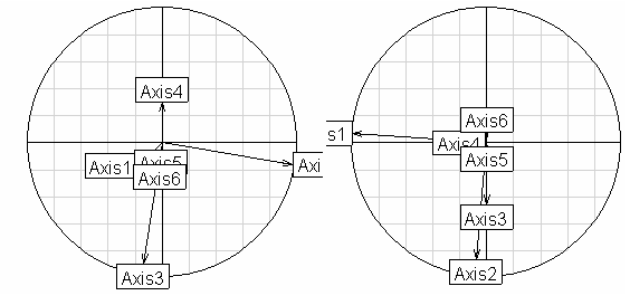
msov
bornes2knb
fac2knb

exemples

```
cfi.coa <- dudi.coa(CFI$tab)
Select the number of axes: 6
cfi.nsc <- dudi.nsc(CFI$tab)
Select the number of axes: 6
sum(cfi.nsc$eig)/59
# [1] 0.1167

# la programmation dans ade4 dans R est un peu différente de celle d'ade4
# classique: on ne retrouve l'indice de diversité de Simpson qu'à un facteur prêt
dudi.nsc.ade4 <- function (df, scannf = TRUE, nf = 2)
{
    df <- data.frame(df)
    lig <- nrow(df)
    col <- ncol(df)
    if (any(df < 0))
        stop("negative entries in table")
    if ((N <- sum(df)) == 0)
        stop("all frequencies are zero")
    row.w <- apply(df, 1, sum)/N
```



```
    col.w <- apply(df, 2, sum)/N
    df <- t(apply(df, 1, function(x) if (sum(x) == 0)
        col.w
    else x/sum(x)))
    df <- sweep(df, 2, col.w)
    df <- data.frame(df)
    X <- as.dudi(df, rep(1, col), row.w, scannf = scannf,
        nf = nf, call = match.call(), type = "nsc")
    X$N <- N
    return(X)
}
cfi.nsc <- dudi.nsc.ade4(CFI$tab)
Select the number of axes: 6

cfi.knb <- bornes2knb(CFI$xy, d = c(0,6,13))
cfi.coa.kmsov <- kmsov(cfi.coa, cfi.knb)
par(mfrow = c(1, 2))
lapply(cfi.coa.kmsov, scatter)
lapply(cfi.coa.kmsov,
 function(x) s.corcircle(x$as, 1, 2))
```



R code

```
"kmsov" <- function(dudi, knb){

require(spdep)
    if (!inherits(dudi, "dudi"))
        stop("object of class 'dudi' expected")
    if (!inherits(knb, "knb"))
        stop("object of class 'knb' expected")

res <- lapply(knb, msov, dudi = dudi)
class(res) <- "kmsov"
return(res)
}
```

7. Variogrammes, co-variogrammes et ordination multi échelles

alias

kmsov.rtest

description

'kmsov.rtest' permet le calcul des variogrammes et co-variogrammes des axes principaux d'une analyse simple suit à une ordination multi échelle.

usage

```
kmsov.rtest(dudi, knb, ax, fac = NULL, nrepet = 300)
```

arguments

- *dudi* : un objet de la classe 'dudi'. Cet objet est le résultat de l'analyse du triplet statistique $(\mathbf{X},\mathbf{Q},\mathbf{D})$
- *knb* : un objet de la classe 'knb'
- *ax* : un vecteur à *k* éléments qui représentent le numéro des axes que l'on souhaite étudier
- *fac* : un facteur. Lorsque l'on réalise cette analyse sur une analyse intra, il faut préciser le facteur afin que les permutations soit faites à l'intérieur des classes seulement
- *nrepet* : un entier représentant le nombre de permutations utilisées pour construite les enveloppes de confiance des variogrammes et co-variogrammes

détails

On note $(\mathbf{X}_C,\mathbf{Q},\mathbf{D})$ le triplet de l'analyse statistique considérée. On note $\mathbf{L}_h$ la matrice de contiguïté du graphe de voisinage *h*. On introduit les matrices $\mathbf{M}_h = \mathbf{D}\mathbf{L}_h\mathbf{D}$ et $\mathbf{N}_h = diag(\mathbf{E}_h)$ avec $\mathbf{E}_h = \mathbf{M}_h\mathbf{1}_n$. On introduit la pondération $K_h = \mathbf{1}_n^t\mathbf{M}_h\mathbf{1}_n$. La matrice de covariance est définie par $\mathbf{Q}^{1/2}\mathbf{X}_C^t\mathbf{D}\mathbf{X}_C\mathbf{Q}^{1/2}$. Ses vecteurs propres définissent les *f* axes principaux $\mathbf{U}_f$. La matrice de covariance pour une classe de distance *h* donnée est définie par $\mathbf{Cov}_h = \mathbf{Q}^{1/2}\mathbf{X}_C^t(\mathbf{N}_h - \mathbf{M}_h)\mathbf{X}_C\mathbf{Q}^{1/2}$. Les variogrammes et co-variogrammes des axes principaux pour une classe de distance *h* sont définis par $\mathbf{F}_h = \frac{\mathbf{U}_f^t\mathbf{Cov}_h\mathbf{U}_f}{K_h}$. Le calcul des variogrammes et co-variogrammes est intimement lié à la décomposition des valeurs propres du schéma de dualité : $\sum_h K_h\mathbf{F}_h = \mathbf{U}_f^t\left(\sum_h \mathbf{Cov}_h\right)\mathbf{U}_f = \mathbf{U}_f^t\mathbf{Cov}\mathbf{U}_f = \boldsymbol{\lambda}$.

valeurs

'kmsov.rtest' retourne un graphique représentant les *k* variogrammes sur la diagonale et les *k*(*k*-1)/2 co-variogrammes sur la partie supérieure.

références

Couteron, P. & Ollier, S. (sous presse) A generalized variogram-based framework for multiscale ordination. Ecology.

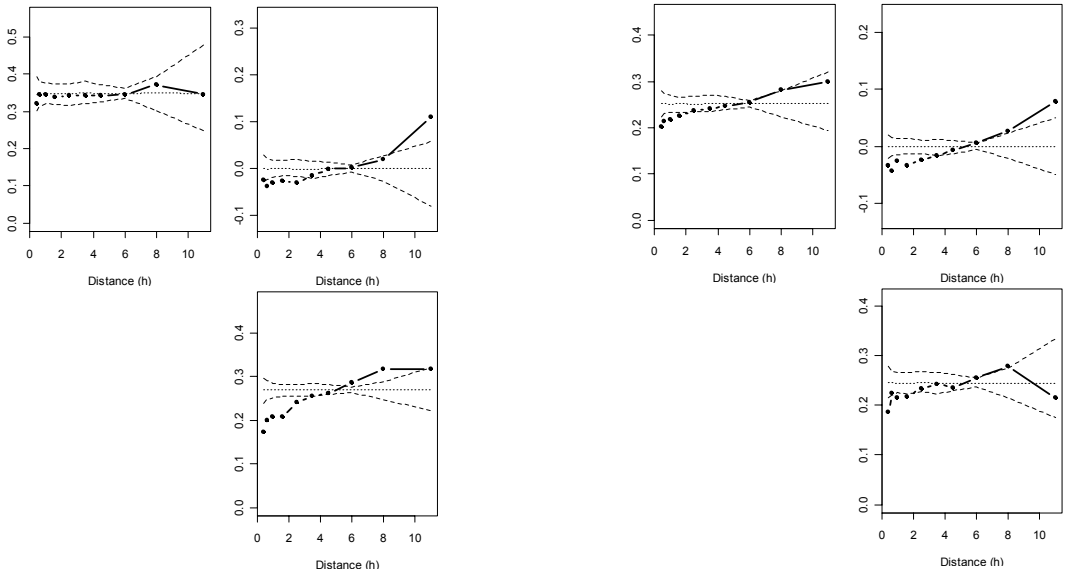
voir également

msov
kmsov
bornes2knb
fac2knb

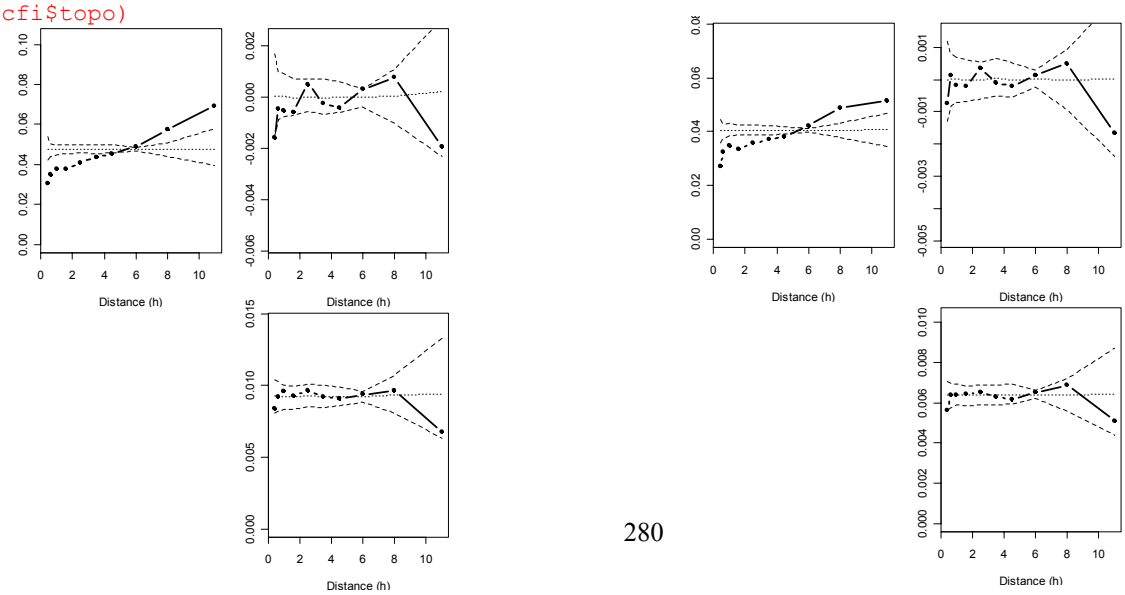
exemples

```
cfi.coa <- dudi.coa(CFI$tab)
Select the number of axes: 6
cfi.nsc <- dudi.nsc(CFI$tab)
Select the number of axes: 6
sum(cfi.nsc$eig)/59
[1] 0.1167
# la programmation dans ade4 dans R est un peu différente de celle d'ade4
# classique: on ne retrouve l'indice de diversité de Simpson qu'à un facteur prêt
dudi.nsc.ade4 <- function (df, scannf = TRUE, nf = 2)
{
    df <- data.frame(df)
    lig <- nrow(df)
    col <- ncol(df)
```

```
    if (any(df < 0))
        stop("negative entries in table")
    if ((N <- sum(df)) == 0)
        stop("all frequencies are zero")
    row.w <- apply(df, 1, sum)/N
    col.w <- apply(df, 2, sum)/N
    df <- t(apply(df, 1, function(x) if (sum(x) == 0)
        col.w
    else x/sum(x)))
    df <- sweep(df, 2, col.w)
    df <- data.frame(df)
    X <- as.dudi(df, rep(1, col), row.w, scannf = scannf,
        nf = nf, call = match.call(), type = "nsc")
    X$N <- N
    return(X)
}
cfi.nsc <- dudi.nsc.ade4(CFI$tab)
Select the number of axes: 6
cfi.coa.with <- within(cfi.coa, CFI$topo)
Select the number of axes: 6
cfi.nsc.with <- within(cfi.nsc, CFI$topo)
Select the number of axes: 6

d <- c(0.38, 0.45, 0.80, 1.20, 2.00, 3.00, 4.00, 5.00, 7.00, 9.00, 13.00)
cfi.knb <- bornes2knb(CFI$xy, d)
cfi.coa.rtest <- kmsov.rtest(cfi.coa, cfi.knb, ax = c(2, 3))
cfi.coa.with.rtest <- kmsov.rtest(cfi.coa.with, cfi.knb, ax = c(2, 3))
```




```
cfi.nsc.rtest <- kmsov.rtest(cfi.nsc, cfi.knb, ax = c(1, 2), fac = cfi$topo)
cfi.nsc.with.rtest <- kmsov.rtest(cfi.nsc.with, cfi.knb, ax = c(1, 2), fac =
cfi$topo)
```



R code

```
"kmsov.rtest" <- function(dudi, knb, ax , fac = NULL, nrepet = 300){

require(spdep)
   if (!inherits(dudi, "dudi"))
        stop("object of class 'dudi' expected")
    if (!inherits(knb, "knb"))
        stop("object of class 'knb' expected")

X <- as.matrix(dudi$tab)
col.w <- dudi$cw
w <- sqrt(col.w)
row.w <- dudi$lw
n <- nrow(X)
p <- ncol(X)
i <- as.matrix(rep(1,n))
nax <- length(ax)
gammafh.lower <- lower.tri(matrix(0, nrow = nax, ncol = nax), diag = T)
Uf <- dudi$c1 * sqrt(dudi$cw)
Uf <- as.matrix(Uf[, ax])

# on calcule pour une relation de voisinage d'une distance h donnée
fun <- function(nb){
gh <- neig2mat(nb2neig(nb))
    # les variogrammes des axes principaux
fun1 <- function(permuter = TRUE){
    if (permuter){
    permutation <- sample(n)
    if(!is.null(fac))
        permutation     <-     unsplit(lapply(split(1:n,     fac),     function(x)
x[sample(length(x))]), fac)
    X <- X[permutation, ]
    row.w <- row.w[permutation]
    mh <- diag(row.w)%*%gh%*%diag(row.w)
    kh <- sum(mh)
    nh <- diag(as.vector(mh%*%i))
    L <- nh-mh
    covh <- t(X)%*%L%*%X
    covh <- sweep(covh, 2, w, "*")
    covh <- covh * w
    gammafh <- t(Uf)%*%covh%*%Uf
    gammafh <- gammafh/kh
    res <- gammafh[gammafh.lower]
    }
    else{
    mh <- diag(row.w)%*%gh%*%diag(row.w)
    kh <- sum(mh)
    nh <- diag(as.vector(mh%*%i))
    L <- nh-mh
    covh <- t(X)%*%L%*%X
    covh <- sweep(covh, 2, w, "*")
    covh <- covh * w
    gammafh <- t(Uf)%*%covh%*%Uf
    gammafh <- gammafh/kh
    res <- gammafh[gammafh.lower]
    }
    return(res)
}
obs <- fun1(permuter = FALSE)
if (nrepet == 0)
    return(t(as.data.frame(obs)))
    else{
    perm <- unlist(lapply(1:nrepet, fun1))
    perm <- matrix(perm, nrow = nrepet, byrow = TRUE)
    res <- rbind.data.frame(obs, perm)
    }
return(res)
}
```

```
res <- lapply(knb, fun)

# on donne la sortie graphique
fun2 <- function(res){
x <- as.numeric(names(res))
plot.default(0, 0, type = "n", asp = 1, xlab = "", ylab = "", xaxt = "n", yaxt =
"n",  xlim  =  c(0,  1),  ylim  =  c(0,  1),  xaxs  =  "i",  yaxs  =  "i",
frame.plot = FALSE)
par(mfrow = c(nax,nax))
par(mar = c(4,3,0.5,0.5))
k <- 1
for(i in 1:nax){
    for(j in i:nax){
        par(mfg = c(i,j))
        y <- unlist(lapply(res, function(x) x[1, k]))
        if (j == i)
            ylim <- c(0, max(y) * 1.5)
            else
                ylim <- c(min(y) * 3, abs(max(y)) * 3)
        plot(x, y, type = "b", pch = 20, lwd = 2, xlab = "Distance (h)", ylab = "",
main = "", ylim = ylim)
        if (nrepet != 0){
            y0975 <- unlist(lapply(res, function(x) quantile(x[2:(nrepet + 1), k],
0.975)))
            ymean <- unlist(lapply(res, function(x) mean(x[2:(nrepet + 1), k])))
            y0025 <- unlist(lapply(res, function(x) quantile(x[2:(nrepet + 1), k],
0.025)))
            lines(x, y0975, lty = 2)
            lines(x, ymean, lty = 3)
            lines(x, y0025, lty = 2)
            }
        k <- k + 1
        }
    }
}
fun2(res)
return(res)
}
```

8. Echelles et graphes de voisinages

alias

bornes2knb
dnearneigh2knb
knearneigh2knb
neig2knb
fac2knb

description

Ces fonctions permettent d'introduire la notion d'échelle par le biais des graphes de voisinage. Elles définissent *k* graphes de voisinage, chacun d'entre eux étant associés à une échelle particulière.

'bornes2knb', 'dnearneigh2knb', 'knearneigh2knb' définissent les graphes à partir de la longueur des arrêtes.
'neig2knb' définit les graphes par le biais de la puissance d'un graphe
'fac2knb' définit les graphes à partir d'une partition de l'espace engendrée par un facteur.

Plusieurs fonctions ('bornes2knb', 'neig2knb', 'fac2knb') génèrent des partitions du graphe complet auxquelles sont associables une décomposition de la variance pour une variable quantitative, et de l'inertie pour un tableau.

usage

```
bornes2knb(xy, d)
dnearneigh(xy, m, l)
knearneigh2knb(xy, v = nrow(xy) - 1)
neig2knb(neig)
fac2knb(fac)
```

arguments

- *xy* : un data.frame dont chaque ligne contient les coordonnées spatiales des points
- *d* : un vecteur dont les éléments définissent les classes de distance considérées
- *m* : un vecteur dont les éléments définissent le centre des intervalles
- *l* : un vecteur dont les éléments définissent la taille des intervalles
- *v* : un vecteur d'entiers correspondant aux nombres de plus proches voisins. Avec l'option par défaut, on obtient le graphe de voisinage complet
- *neig* : un objet de la classe 'neig' (seul un graphe connexe est acceptable)
- *fac* : un facteur dont les modalités définissent une partition de l'espace
- *kfac* : un objet de la classe 'kfac'

détails

On note $\mathbf{M}_k$ la matrice de contiguïté du graphe *k*. On a alors $\mathbf{M}_k(a,b)=1$ si et seulement *a* et *b* sont voisins.
'bornes2knb' : deux points *a* et *b* sont considérés comme voisins si et seulement si la distance qui les sépare est comprise entre d_k et d_{k+1}.
'dnearneigh2knb : deux points *a* et *b* sont considérés comme voisins si et seulement si la distance qui les sépare est comprise entre $m_k - \frac{l_k}{2}$ et $m_k + \frac{l_k}{2}$.
'knearneigh2knb' : deux points *a* et *b* sont considérés comme voisins si et seulement si *b* fait partie des v_k plus proches voisins de a.
'neig2knb' : deux points *a* et *b* sont considérés comme voisins au niveau *k* si la longueur du chemin de longueur minimum qui les relie vaut exactement *k*.
fac2knb : deux points *a* et *b* sont considérés comme voisins si ils portent la même modalité.

valeurs

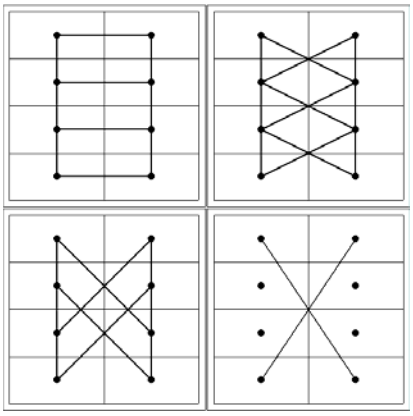
Ces fonctions retournent des objets de la classe 'knb' : il s'agit d'une liste d'objets de la classe 'nb'.

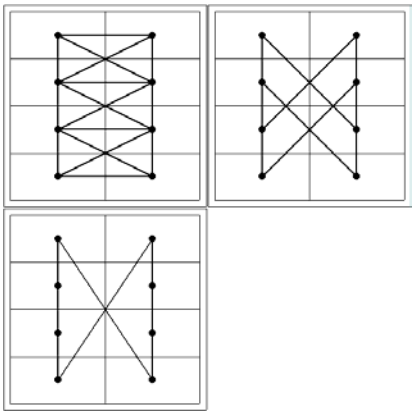
références

Benali, H. & Escofier, B. (1990) Analyse factorielle lissée et analyse factorielle des différences locales. Revue de Statistique Appliquée, 38, 55-76.
Smouse, P. & Peakall, R. (1999) Spatial autocorrelation analysis of individual multiallele and multilocus genetic structure. Heredity, 82, 561-573.

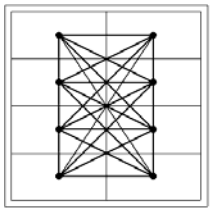
exemples

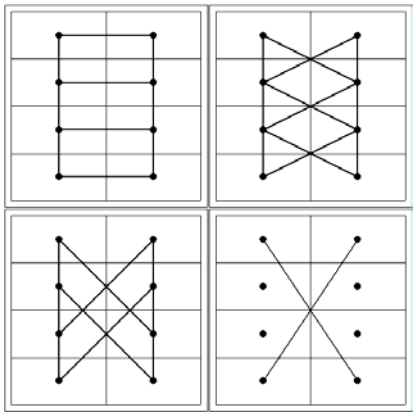
```
grille <- gridrowcol(4, 2)
fun <- function(x){
plot(grille$xy, xlim = c(0.5,2.5), ylim = c(0.5,4.5),
 type = "n", axes = F, xlab = "", ylab   = "")
s.label(grille$xy, area = grille$area, neig = x,
 cneig = 2, clab = 0, cpoint = 3, inc = FALSE,
 addaxes = FALSE, grid = FALSE, add.plot = TRUE)
}

# bornes2knb
d <- c(0, 1.1, 2.1, 3.1, 4.1)
knb <- bornes2knb(grille$xy, d)
kneig <- lapply(knb, nb2neig)
summary(knb)
     Length Class Mode
d_0.55 8      nb    list
d_1.6  8      nb    list
d_2.6  8      nb    list
d_3.6  8      nb    list

summary(kneig)
      Length Class Mode
d_0.55 20     neig  numeric
d_1.6  20     neig  numeric
d_2.6  12     neig  numeric
d_3.6   4     neig  numeric

par(mfrow = c(2, 2))
par(mar = c(0,0,0,0))
lapply(kneig, fun)

u <- lapply(kneig, neig2mat)
v <- u[[1]]
for(i in 2:length(u)) v <- v + u[[i]]
v
  1 2 3 4 5 6 7 8
1 0 1 1 1 1 1 1 1
2 1 0 1 1 1 1 1 1
3 1 1 0 1 1 1 1 1
4 1 1 1 0 1 1 1 1
5 1 1 1 1 0 1 1 1
6 1 1 1 1 1 0 1 1
7 1 1 1 1 1 1 0 1
8 1 1 1 1 1 1 1 0

# dnearneigh2knb
m <- c(1, 2, 3)
l <- rep(1, 3)
knb <- dnearneigh2knb(grille$xy, m, l)
kneig <- lapply(knb, nb2neig)
summary(knb)
    Length Class Mode
m_1 8      nb    list
m_2 8      nb    list
m_3 8      nb    list

summary(kneig)
    Length Class Mode
m_1 32     neig  numeric
m_2 16     neig  numeric
m_3  8     neig  numeric

par(mfrow = c(2, 2))
```



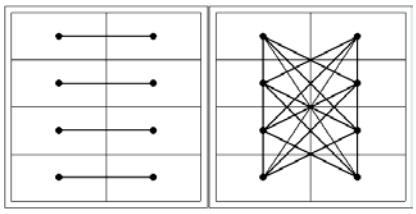


```
par(mar = c(0,0,0,0))
lapply(kneig, fun)

u <- lapply(kneig, neig2mat)
v <- u[[1]]
for(i in 2:length(u)) v <- v + u[[i]]
v
  1 2 3 4 5 6 7 8
1 0 1 1 1 1 1 1 1
2 1 0 1 1 1 1 1 1
3 1 1 0 1 1 1 1 1
4 1 1 1 0 1 1 1 1
5 1 1 1 1 0 1 1 1
6 1 1 1 1 1 0 1 1
7 1 1 1 1 1 1 0 1
8 1 1 1 1 1 1 1 0

# knearneigh2knb
knb <- knearneigh2knb(grille$xy)
kneig <- lapply(knb, nb2neig)
summary(knb)
    Length Class Mode
v_7 8      nb    list

summary(kneig)
   Length Class Mode
v_7 56     neig  numeric

par(mfrow = c(1, 1))
par(mar = c(0,0,0,0))
lapply(kneig, fun)

u <- lapply(kneig, neig2mat)
v <- u[[1]]
v
  1 2 3 4 5 6 7 8
1 0 1 1 1 1 1 1 1
2 1 0 1 1 1 1 1 1
3 1 1 0 1 1 1 1 1
4 1 1 1 0 1 1 1 1
5 1 1 1 1 0 1 1 1
6 1 1 1 1 1 0 1 1
7 1 1 1 1 1 1 0 1
8 1 1 1 1 1 1 1 0


# neig2knb
knb <- neig2knb(grille$neig)
kneig <- lapply(knb, nb2neig)
summary(knb)
       Length Class Mode
l.min_1 8      nb    list
l.min_2 8      nb    list
l.min_3 8      nb    list
l.min_4 8      nb    list

summary(kneig)
       Length Class Mode
l.min_1 20     neig  numeric
l.min_2 20     neig  numeric
l.min_3 12     neig  numeric
l.min_4  4     neig  numeric

par(mfrow = c(2, 2))
par(mar = c(0,0,0,0))
lapply(kneig, fun)

u <- lapply(kneig, neig2mat)
```





```
v <- u[[1]]
for(i in 2:length(u)) v <- v + u[[i]]
v
  1 2 3 4 5 6 7 8
1 0 1 1 1 1 1 1 1
2 1 0 1 1 1 1 1 1
3 1 1 0 1 1 1 1 1
4 1 1 1 0 1 1 1 1
5 1 1 1 1 0 1 1 1
6 1 1 1 1 1 0 1 1
7 1 1 1 1 1 1 0 1
8 1 1 1 1 1 1 1 0

# fac2knb
fac <- as.factor(rep(1:4, rep(2, 4)))
knb <- fac2knb(fac)
kneig <- lapply(knb, nb2neig)
summary(knb)
      Length Class Mode
intra 8      nb    list
inter 8      nb    list

summary(kneig)
      Length Class Mode
intra  8     neig  numeric
inter 48     neig  numeric

par(mfrow = c(2, 2))
par(mar = c(0,0,0,0))
lapply(kneig, fun)

u <- lapply(kneig, neig2mat)
v <- u[[1]]
for(i in 2:length(u)) v <- v + u[[i]]
v
  1 2 3 4 5 6 7 8
1 0 1 1 1 1 1 1 1
2 1 0 1 1 1 1 1 1
3 1 1 0 1 1 1 1 1
4 1 1 1 0 1 1 1 1
5 1 1 1 1 0 1 1 1
6 1 1 1 1 1 0 1 1
7 1 1 1 1 1 1 0 1
8 1 1 1 1 1 1 1 0
```



R code

```
"bornes2knb" <- function(xy, d){
require(spdep)

xy <- as.matrix(xy)
nnb <- length(d) - 1
res <- NULL
u <- NULL
for(i in 1:nnb){
    res[[i]] <- dnearneigh(xy, d[i], d[i + 1])
    u[i] <- (d[i] + d[i + 1]) / 2
    }
v <- rep("d", nnb)
u <- paste(v, u, sep = "_")
names(res) <- u
class(res) <- "knb"
return(res)
}

####   ####   ####   ####
####   ####   ####   ####

"dnearneigh2knb" <- function(xy, m, l){
```

```
require(spdep)

xy <- as.matrix(xy)
nnb <- length(m)
res <- NULL
u <- NULL
for(i in 1:nnb)
    res[[i]] <- dnearneigh(xy, m[i] - l[i] / 2, m[i] + l[i] / 2)

u <- rep("m", nnb)
u <- paste(u, m, sep = "_")
names(res) <- u
class(res) <- "knb"
return(res)
}

####   ####   ####   ####
####   ####   ####   ####

"knearneigh2knb" <- function(xy, v = nrow(xy) - 1){
require(spdep)

xy <- as.matrix(xy)
nnb <- length(v)
res <- NULL
u <- NULL
for(i in 1:nnb)
    res[[i]] <- knn2nb(knearneigh(xy, v[i], lonlat = FALSE))

u <- rep("v", nnb)
u <- paste(u, v, sep = "_")
names(res) <- u
class(res) <- "knb"
return(res)
}

####   ####   ####   ####
####   ####   ####   ####

"neig2knb" <- function(neig){
require(spdep)

    l.min <- function(neig){
        # fonction technique: renvoie la matrice qui donne la distance entre 2
       points
        # comme longueur minimum du chemin qui les relie
        x <- neig.util.LtoG(neig)
        x <- as.matrix(x) ; n <- nrow(x)
        auxi1 <- x ; auxi2 <- x
        m <- rep(0, n) ; y <- x ; m[1] <- sum(x);
        for (itour in 2:n) {
            auxi2 <- auxi2%*%auxi1 ; diag(auxi2) <- 0 ;
            auxi2 <- (x == 0)*auxi2 ; auxi2 <- (auxi2>0)*itour
            if (sum(auxi2) == 0) break
            x <- x+auxi2
        }
        return(x)
    }
x <- l.min(neig)
n <- nrow(x)
nnb <- max(x)
res <- list(neig)
    for (i in 1:nnb){
        M <- (x == i) * 1
        res[[i]] <- neig2nb(neig(mat01 = M))
    }
u <- rep("l.min", nnb)
u <- paste(u, 1:nnb, sep = "_")
```

```
names(res) <- u
class(res) <- "knb"
return(res)
}

####   ####   ####   ####
####   ####   ####   ####

"fac2knb" <- function(fac){
require(spdep)
tab <- table(fac)
l <- length(tab)
m <- levels(fac)
n <- sum(tab)
res <- matrix(0, nrow = n, ncol = n)
for(i in 1:l){
    u <- (1:n)[fac == m[i]]
    res[u,u] <- matrix(1, nrow = tab[i], ncol = tab[i])
    }
diag(res) <- rep(0, n)
res1 <- neig2nb(neig(mat01 = res))
mat <- as.vector(rep(1, n))%*%t(as.vector(rep(1, n)))
diag(mat) <- rep(0, n)
res <- mat - res
res2 <- neig2nb(neig(mat01 = res))
res <- list(res1, res2)
names(res) <- c("intra", "inter")
class(res) <- "knb"
return(res)
}
```

9. Base orthonormée et famille de projecteurs

alias

circ2level
wavelet2level
taxo2level
phylog2level

description

Ces fonctions définissent une partition de l'ensemble des vecteurs de la base orthonormée. A chaque élément de la partition est associé un projecteur. On associe donc ensuite à une partition et une base orthonormée, une famille de *k* projecteurs.

usage

```
circ2level(n)
wavelet2level(n)
taxo2level(taxo)
phylog2level(phylog)
```

arguments

- *n* : est un entier correspondant au nombre d'unités statistiques
- *taxo* : est un objet de la classe 'taxo', sous-classe de la classe 'phylog'
- *phylog* : est un objet de la classe 'phylog'

détails

Soit $A=\{1,2,...,n\}$ l'ensemble des entiers compris entre *1* et *n*. Les fonctions définissent une partition de cet ensemble : $A=A_1\cup A_2\cup ...\cup A_k$. Soit $\mathbf{B}$ une base orthonormée. On définit alors les matrices $\mathbf{B}_j$ par l'ensemble des vecteurs b^i de $\mathbf{B}$ tel que $i\in A_j$. Le projecteur associé à la base $\mathbf{B}_j$ est défini par $\mathbf{\Pi}_j=\mathbf{B}_j\mathbf{B}_j^t$.

'circ2level' introduit la partition qui permet de définir à partir des bases orthonormées associées aux graphes circulaires, les projecteurs permettant de calculer le periodogram. A chaque projecteur est alors associé une fréquence particulière.

'wavelet2level' introduit la partition qui permet de définir à partir des bases orthonormées associées aux ondelettes, les projecteurs permettant de calculer le scalogram. Pour l'instant, les fonctions associées aux ondelettes ne permettent pas de travailler avec des variables dont le nombre d'unités statistiques n'est pas une puissance de 2.

'taxo2level' introduit la partition qui permet de définir à partir des bases orthonormées associées aux taxonomies, les projecteurs permettant de calculer la décomposition de la variance par niveau taxonomique.

'phylog2level' est une généralisation de 'taxo2level' à une phylogénie. Elle introduit la partition qui permet de définir à partir des bases orthonormées associées aux phylogénies, les projecteurs permettant de calculer la décomposition de la variance par nœuds.

On peut également définir une partition quelconque. En particulier, on peut choisir de définir un projecteur pour chaque vecteur propre (*k* = *n*). On a alors une stricte équivalence en la décomposition de la variance par *k* formes bilinéaires symétriques et la décomposition par les vecteurs d'une base orthonormée.

valeurs

Ces fonctions retournent un facteur à *k* modalités qui représente la partition de l'ensemble des entiers compris entre *1* et *n*.

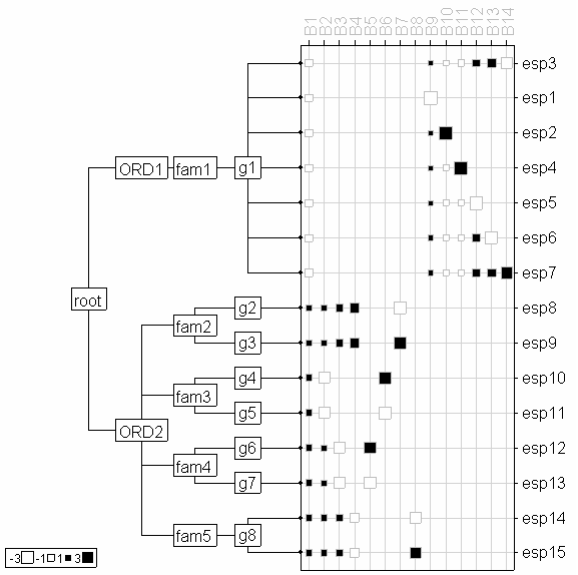
voir également

orthobasis2kfbs
mld

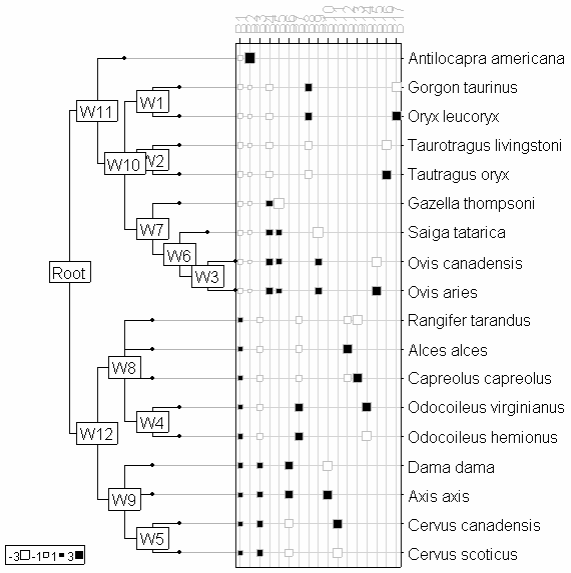
exemples

```
circ2level(4)
[1] A A B
Levels: A B
```

```
circ2level(5)
[1] A A B B
Levels: A B
wavelet2level(5)
Error in wavelet2level(5) :
 n is not a power of 2
wavelet2level(8)
[1] B3 B2 B2 B1 B1 B1 B1
Levels: B3 B2 B1
taxo <- as.taxo(taxo.eg[[1]])
taxo.phy <- taxo2phylog(taxo)
table.phylog(taxo.phy$Bscores,
 taxo.phy, clabel.nod = 1)
taxo2level(taxo)
[1] A B B B C C C D D D D D D D
Levels: A B C D
phylog2level(taxo.phy)
[1] root ORD2 ORD2 ORD2 fam4 fam3
 fam2 g8   g1   g1   g1   g1   g1   g1
Levels: fam2 fam3 fam4 g1 g8 ORD2 root
```




```
ung.phy <- newick2phylog(ungulates$tre)
table.phylog(ung.phy$Bscores, ung.phy,
 clabel.nod = 1)
phylog2level(ung.phy)
[1] Root W11  W12  W10  W7   W6   W9   W8
   W10  W3   W5   W4   W2   W1   W9
[16] W8   W8
Levels: Root W1 W10 W11 W12 W2 W3 W4
 W5 W6 W7 W8 W9
```



R code

```
"circ2level" <- function(n){
if (floor(n/2) == floor((n+1)/2)){
    # n is even
    u <- 1:(n/2) # u <- LETTERS[1:(n/2)]
    v <- rep(u, c(rep(2, n/2-1), 1))
    res <- as.factor(v)
    }
        else{
        # n is odd
        u <- 1:floor(n/2) # LETTERS[1:floor(n/2)]
        v <- rep(u, rep(2, floor(n/2)))
        res <- as.factor(v)
        }
return(res)
}

####   ####   ####   ####
####   ####   ####   ####

"wavelet2level" <- function(n){
# on vérifie que n = 2**J avec J le nombre de niveau
J <- log(n)/log(2)
b <- floor(J)
if ((J-b)^2 > 1e-10) stop ("n is not a power of 2")

#on construit les J niveaux de décomposition
u <- LETTERS[1:J]
v <- rep(2,J)**(0:(J-1))
level <- rep(u, v)
level <- as.factor(level)
```

```
w <- paste(rep("B", J), J:1, sep = "")
levels(level) <- w
return(level)
}

####   ####   ####   ####
####   ####   ####   ####

"taxo2level" <- function(taxo){
if (class(taxo)[2] != "taxo") stop(" 'taxo' expected with class taxo ")

J <- ncol(taxo) + 1
u <- LETTERS[1:J]

fac <- taxo[,J-1]
v <- c(nlevels(fac) -1)
for(i in 2:(J-1))    v[i] <- sum(apply(table(taxo[,(J-i+1)], taxo[,(J-i)]) != 0, 1,
sum) - 1)
fac <- taxo[,1]
v[J] <- sum(table(fac) - 1)

res <- rep(u, v)
res <- as.factor(res)
return(res)
}

####   ####   ####   ####
####   ####   ####   ####

"phylog2level" <- function(phylog){
if (class(phylog) != "phylog") stop(" 'phylog' expected with class phylog ")

Blab <- phylog$Blabels
l <- length(Blab)
Bnames <- names(Blab)
u <- sapply(Blab, function(x) strsplit(x, "/"))
v <- unlist(lapply(u, function(x) sum(x == "x")))
v <- (1:l)[v == 0]
Blab <- Blab[v]
l <- length(Blab)
Bnames <- Bnames[v]
u <- u[v]
u <- lapply(u, as.numeric)
res <- rep(Bnames, unlist(lapply(u, length)))
res <- res[order(unlist(u))]
res <- as.factor(res)
return(res)
}
```

10. Décomposition d'une variable à différentes échelles

alias

mld

description

'mld' est une fonction qui assure la décomposition d'une variable sur un ensemble de vecteurs d'une base orthonormée.

usage

```
mld(x, orthobas, level, na.action = c("fail", "mean"), plot = TRUE, dfxy = NULL,
phylog = NULL, ...)
```

arguments

- *x* : est un vecteur correspondant à une variable
- *orthobas* : est un objet de la classe 'orthobasis'
- *level* : est un facteur
- *na*.action : est une chaîne de caractères désignant l'action à mettre en oeuvre si la variable présente des données manquantes.
- *plot* : est une variable binaire. Si TRUE, la fonction retourne une représentation graphique des vecteurs de la décomposition
- *dfxy* : est un data frame donnant les coordonnées d'un semi de point
- *phylog* : est un objet de la classe 'phylog'
- … : arguments supplémentaires

détails

Chaque variable $\tilde{\mathbf{x}}$ admet une décomposition unique sur les vecteurs de la base **B** :

$$\tilde{\mathbf{x}} = \sum_{i=1}^{n-1} r_i \mathbf{b}_i \cdot$$

Si l'on note $\mathbf{B}_E$, la matrice constituée par l'ensemble E des card(E) vecteurs de **B**, toute partition $E_1 \cup E_2 \cup ... \cup E_K = \{1, 2, ..., n-1\}$ définit une décomposition de la variable en K composantes orthogonales par :

$$\tilde{\mathbf{x}} = \sum_{k=1}^{K} \tilde{\mathbf{x}}_k = \sum_{k=1}^{K} \sum_{i}^{card(E_k)} r_i \mathbf{b}_i$$

valeurs

'mld' retourne les K composantes.

références

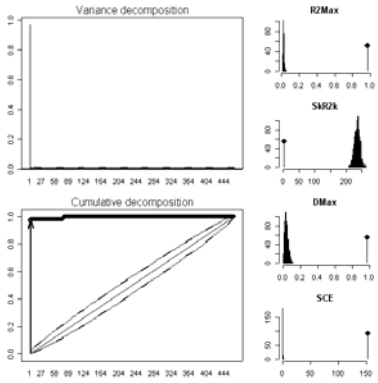
Percival, D.B. & Walden, A.T. (2000) Wavelet Methods for Time Series Analysis Cambridge University Press.

voir également

orthobasis
orthogram
wavelet2level
circ2level
taxo2level
phylog2level
mra (programathèque waveslim)

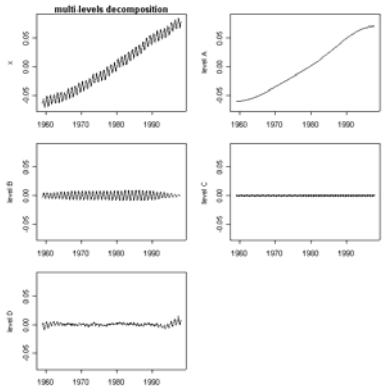
exemples

```
# exemple d'une série temporelle
data(co2)
help(co2)
x <- log(co2)
co2 <- as.data.frame(co2)
names(co2) <- "co2"
dotchart.line(co2, cdot = 0, axe1 = F,
 joining = FALSE)
orthobas <- orthobasis.line(length(x))
orthogram(x, orthobas, high.scores = 7)
$w
class: krandtest
test number:   4
permutation number:   999
  test   obs      P(X<=obs) P(X>=obs)
```

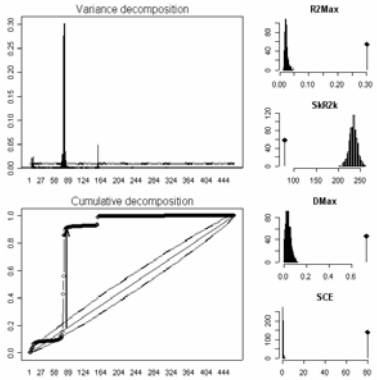


```
1 R2Max 0.968    1          0.001
2 SkR2k 2.641    0.001      1
3 Dmax  0.973    1          0.001
4 SCE   152.243 1           0.001

$scores.order
[1]   1  79   2  77   3  78 156

Warning message:

# Quand il y a une tendance, elle représente toute la variabilité.
# Le test sur les vecteurs 79 et 77, et 156 ne sont donc pas significatif alors que
# les résidus sont structurés : le problème multiéchelle n'est pas facile
```

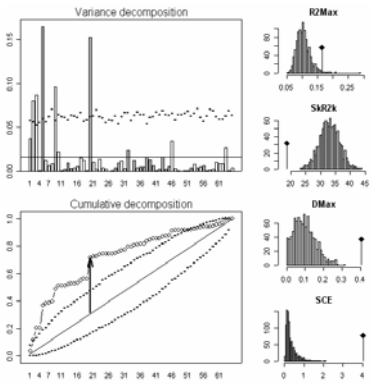
```
level <- rep("D", 467)
level[1:3] <- rep("A", 3)
level[c(77,78,79,81)]<-rep("B", 2)
level[156] <- "C"
level <- as.factor(level)
res <- mld(x, orthobas, level)
```




```
# on travaille sur les résidus
e <- res[,2]+res[,3]+res[,4]
orthogram(e, orthobas, high.scores = 8)
$w
class: krandtest
test number:   4
permutation number:   999
  test   obs     P(X<=obs) P(X>=obs)
1 R2Max 0.301  1          0.001
2 SkR2k 78.984 0.001      1
3 Dmax  0.729  1          0.001
4 SCE   80.221 1          0.001

$scores.order
[1]  79  77  78 156  81  75   9   5
```




```
# Les deux fonctions orthogram et mld se complètent très bien. Elles sont
# l'expression d'une classe de méthodes appelée transformations orthonormales.


# exemple d'un trait biologique associé à une phylogénie
vfruit <- palm$traits$vfruit
vfruit <- scalewt(vfruit)
palm.phy <- newick2phylog(palm$tre)
orthogram(vfruit, phylog = palm.phy,
 high.scores = 10)
$w
class: krandtest
test number:   4
permutation number:   999
  test   obs     P(X<=obs) P(X>=obs)
1 R2Max 0.165  0.983      0.019
2 SkR2k 18.623 0.001      1
3 Dmax  0.406  1          0.001
4 SCE   4.038  1          0.001

$scores.order
 [1]  5 20  9  3  2  1 46 63 32 10
```



```
u <- cbind.data.frame(vfruit,
 palm.phy$Bscores[,c(3,4,6,13,21)])
row.names(u) <- row.names(palm$traits)
names(u) <- c("vfruit","B3","B4",
 "B6","B13","B21")
table.phylog(u, palm.phy,
 labels.nod = palm.phy$Blabels,
 clabel.nod = 0.7, f.phylog = 0.8, csize = 2,
 clabel.row = 0.7, clabel.col = 0.7)


# la même chose avec mld
level <- rep("F", 65)
level[c(4, 21, 3, 6, 13)] <- LETTERS[1:5]
level <- as.factor(level)
res <- mld(as.vector(vfruit),
 palm.phy$Bscores, level, phylog = palm.phy,
 clabel.nod = 0.7, f.phylog=0.8, csize=2,
 clabel.row=0.7,clabel.col=0.7)
```

R code

```
"mld"<- function (x, orthobas, level, na.action = c("fail", "mean"), plot=TRUE,
dfxy = NULL, phylog = NULL,  ...)
{

# on fait les vérifications sur x
if (!is.numeric(x))
        stop("x is not numeric")
nobs <- length(x)
if (any(is.na(x))) {
        if (na.action == "fail")
            stop(" missing values in 'x'")
        else if (na.action == "mean")
            x[is.na(x)] <- mean(na.omit(x))
        else stop("unknown method for 'na.action'")
    }

# on fait les vérifications sur orthobas :
# class, dimension, orthogonalité, orthonormalité
if (!inherits(orthobas, "data.frame")) stop ("'orthobas' is not a data.frame")
    if (nrow(orthobas) != nobs) stop ("non convenient dimensions")
    if      (ncol(orthobas)      !=      (nobs-1))      stop      (paste("'orthobas'
has",ncol(orthobas),"columns, expected:",nobs-1))

vecpro <- as.matrix(orthobas)
npro <- ncol(vecpro)

w <- t(vecpro/nobs)%*%vecpro
    if (any(abs(diag(w)-1)>1e-07)) {
        stop("'orthobas' is not orthonormal for uniform weighting")
    }

diag(w) <- 0
    if ( any( abs(as.numeric(w))>1e-07) ) stop("'orthobas' is not orthogonal for
uniform weighting")

# on calcule les différents vecteurs associés
# à la décomposition orthonormale de la variable

    # si x n'est pas centrée, on la centre pour la pondération uniforme
    if (mean(x)!=0)
            x <- x-mean(x)

    # on calcul les coefficients de corrélation
    # entre la variable et les vecteurs de la base
    coeff <- t(vecpro/nobs)%*%as.matrix(x)
```

```
    # on calcul les vecteurs associés à la décomposition et au facteur level
    if (!is.factor(level))
            stop("'level' is not a factor")
            if (length(level) != (nobs-1))
                    stop         (paste("'level'         has",length(level),"values,
expected:",nobs-1))
    res <- matrix(0, nrow = nobs, ncol = nlevels(level))
    coeff <- split(coeff, level)
    vecpro <- as.data.frame(t(vecpro))
    vecpro <- split(vecpro, level)
    for (i in 1:nlevels(level))
        res[,i] <- t(vecpro[[i]])%*%as.matrix(coeff[[i]])
    res <- as.data.frame(res)
    names(res) <- paste("level", levels(level), sep=" ")


# on fait les sorties graphiques si elles sont demandées:
# c'est pas parfait mais c'est pour donner une idée
if (plot==TRUE){
    # rajouter les données circulaires
    if (is.ts(x)){
        # pour les séries temporelles
        u <- attributes(x)$tsp
        tab <- ts(res, start = u[1], end = u[2], frequency = u[3])
        tab <- ts.union(x, tab)
        u <- range(tab)
        opar <- par(mfrow = par("mfrow"), mar = par("mar"))
        on.exit(par(opar))
        mfrow <- n2mfrow(nlevels(level)+1)
        par(mfrow = mfrow)
        par(mar = c(2.5, 5, 1.5, 0.6))
        plot.ts(x, ylim = u, ylab = "x", main = "multi-levels decomposition")
        for (i in 1:nlevels(level))
                plot(tab[,i+1], ylim = u, ylab = names(res)[i], main = "")
        }

    if (is.vector(x)){
        if (!is.null(dfxy)){
            # pour les données 2 D
            opar <- par(mfrow = par("mfrow"), mar = par("mar"))
            on.exit(par(opar))
            mfrow <- n2mfrow(nlevels(level)+1)
            par(mfrow = mfrow)
            par(mar = c(0.6, 2.6, 0.6, 0.6))
            s.value(dfxy, x, sub = "x", ...)
            for (i in 1:nlevels(level))
                if (max((1:(nobs-1))[level == levels(level)[i]])<(nobs/2)){
                    s.image(dfxy, res[,i])
                    s.value(dfxy, res[,i], sub = names(res)[i], add.plot=TRUE, ...)
                    }
                else
                    s.value(dfxy, res[,i], sub = names(res)[i], ...)
            }
        else {
            if (!is.null(phylog)){
                # pour les données associées à une phylogénie
                tab <- cbind.data.frame(x, res)
                row.names(tab) <- names(phylog$leaves)
                table.phylog(tab, phylog, ...)
                }
            else {
                # pour les transects
                par(mfrow = c(nlevels(level)+1,1))
                par(mar = c(2, 5, 1.5, 0.6))
                u <- range(cbind(x, res))
                w <- trunc(u)
                w <- c(w[1],0,w[2])
```

```
                plot(x, type="h", ylim = u, axes = FALSE, ylab = "x", main =
"multi-levels decomposition")
                axis(side = 2, at = w, labels = as.character(w))
                for (i in 1:nlevels(level)){
                    plot(res[,i], type="h", ylim = u, axes = FALSE, ylab =
names(res)[i], main = "")
                    axis(side = 2, at = w, labels = as.character(w))
                    }
                v <- seq(0, nobs, by = (nobs/10))
                axis(side=1, at = v, labels = as.character(v))
                }
            }
        }
    }
return(res)
}
```

11. Ordination sous contrainte spatiale version Geary

alias

msov

description

'msov' est une fonction qui généralise l'analyse de la variance locale d'une variable par l'indice de Geary à l'analyse d'un tableau multivarié. On introduit la contrainte spatiale au sein de l'analyse d'un triplet statistique par l'intermédiaire d'un graphe de voisinage. Cette approche est différente de celle proposée dans Thioulouse où les auteurs cherchent à réconcilier les deux points de vue Geary-Moran. Ici l'objectif est de généraliser l'ordination multiéchelle sous sa forme variogramme à l'ensemble des analyses à 1 tableau, en conservant l'inertie de l'analyse simple. Dans certains cas, on aboutit alors une décomposition de la diversité.
'summary.msov' est une fonction donnant un résumé de l'analyse

usage

```
msov(dudi, nb, scannf = FALSE, nf = 6)
```

arguments

- *dudi* : un objet de la classe 'dudi'. Cet objet est le résultat de l'analyse du triplet statistique $(\mathbf{X},\mathbf{Q},\mathbf{D})$
- *nb* : un objet de la class 'nb' représentant le graphe de voisinage considéré
- *scannf* : une variable binaire. Si TRUE, le nombre de valeurs propres à retenir est demandé à l'utilisateur. Sinon, ce nombre égale l'argument *nf*
- *nf* : un entier désignant le nombre de valeurs propres à conserver si *scannf* = FALSE

détails

On note $(\mathbf{X}_C,\mathbf{Q},\mathbf{D})$ le triplet de l'analyse statistique considérée. L'analyse de base passe par la décomposition en valeur singulière (svd) du tableau $\mathbf{Q}^{1/2}\mathbf{X}_C^t\mathbf{D}^{1/2}$, ou par l'analyse (eigenanalysis) de la matrice $\mathbf{Q}^{1/2}\mathbf{X}_C^t\mathbf{D}\mathbf{X}_C\mathbf{Q}^{1/2}$. Les vecteurs propres obtenus maximisent l'inertie projetée et assurent une décomposition optimale de l'inertie du tableau : $tr\left(\mathbf{Q}^{1/2}\mathbf{X}_C^t\mathbf{D}\mathbf{X}_C\mathbf{Q}^{1/2}\right)=\sum_{i=1}^{r}\lambda_i$.

On note $\mathbf{L}$ la matrice de contiguïté du graphe de voisinage. On introduit les matrices $\mathbf{M}=\mathbf{DLD}$ et $\mathbf{N}=diag(\mathbf{E})$ avec $\mathbf{E}=\mathbf{M}\mathbf{1}_n$. L'analyse sous contrainte est définie comme l'analyse du triplet $(\mathbf{X}_C,\mathbf{Q},\mathbf{N}-\mathbf{M})$. En particulier, si $\mathbf{L}$ est la matrice de contiguïté du graphe de voisinage considérant tous les points comme voisin, l'analyse du triplet $(\mathbf{X}_C,\mathbf{Q},\mathbf{N}-\mathbf{M})$ est équivalente à l'analyse du triplet $(\mathbf{X}_C,\mathbf{Q},\mathbf{D})$. Cette propriété est directement liée aux deux écritures de la variance à savoir $\mathrm{var}(\mathbf{x})=\sum p_i\left(x_i-\bar{x}\right)^2=\mathbf{x}_c^t\mathbf{D}\mathbf{x}_c$ et $\mathrm{var}(\mathbf{x})=\frac{1}{2}\sum p_i p_j\left(x_i-x_j\right)^2=\mathbf{x}\left(\mathbf{N}-\mathbf{M}\right)\mathbf{x}=\mathbf{x}_c^t\left(\mathbf{N}-\mathbf{M}\right)\mathbf{x}_c$. Cette propriété se généralise à l'inertie d'un tableau en générale.

Les axes principaux de l'analyse simple maximise la variance projetée et n'ont aucune propriété quant à leur structure spatiale. On peut alors chercher ceux qui maximise la variance locale. On peut partir du coefficient de contiguité de Geary pour définir le critère à maximiser: $c(\mathbf{XQv})=\frac{\mathbf{v}^t\mathbf{QX}^t\left(\mathbf{N}-\mathbf{M}\right)\mathbf{XQv}}{\mathbf{v}^t\mathbf{QX}^t\mathbf{DXQv}}$. Considérons alors la matrice $\mathbf{H}=\mathbf{X}^t\left(\mathbf{N}-\mathbf{M}\right)\mathbf{XQ}$. Elle est $\mathbf{Q}$-symétrique et possède une base de vecteurs propres $\mathbf{Q}$-orthonormés. Le premier vecteur propre $\mathbf{v}_1$ associé à la plus grande valeur propre λ_1 maximise sous contrainte d'orthonormalité $\|\mathbf{v}\|_\mathbf{Q}=1$ la quantité $\langle\mathbf{Hv}|\mathbf{v}\rangle_\mathbf{Q}=\mathbf{v}^t\mathbf{QX}^t\left(\mathbf{N}-\mathbf{M}\right)\mathbf{XQv}=\|\mathbf{XQv}\|_\mathbf{D}^2\,c(\mathbf{XQv})$. On maximise donc un compromis entre l'inertie projetée et la variance locale des coordonnées exprimée par le coefficient de contiguité de Geary.

La fonction 'summary.msov' donne d'une part un résumé des statistiques d'inertie associées à l'analyse simple. La colonne 'eig' correspond aux valeurs propres, qui donnent l'inertie projetée, c'est-à-dire la variance des coordonnées sur les axes principaux $\|\mathbf{XQv}\|_{\mathbf{D}}^2$. La colonne 'cum' donne la somme cummulée. La colonne ratio donne le pourcentage d'inertie expliquée. La colonne 'geary' donne le coefficient de contiguité des coordonnées sur chaque axe $c(\mathbf{XQv})$. Elle donne en suite les statistiques associées à l'analyse sous contrainte. La colonne 'eig' donnent les valeurs propres. Cette fois-ci les valeurs propres correspondent à la variance locale des coordonnées sur chaque axe $\mathbf{v}'\mathbf{QX}'(\mathbf{N}-\mathbf{M})\mathbf{XQv} = c(\mathbf{XQv})\|\mathbf{XQv}\|_{\mathbf{D}}^2$. Les colonnes 'cum' est 'ratio' sont définies comme précédemment. La colonne 'var' donne la variance des coordonnées sur chaque axe conservé $\|\mathbf{XQv}\|_{\mathbf{D}}^2$. La colonne 'geary' donne le coefficient de contiguité des coordonnées $c(\mathbf{XQv})$.

valeurs

'msov' retourne un objet de la classe 'msov', sous classe de la classe 'dudi'

références

Couteron, P. & Ollier, S. (sous presse) A generalized variogram-based framework for multiscale ordination. Ecology.
Wagner, H.H. (2003) Spatial covariance in plant communities: integrating ordination, geostatistics, and variance testing. Ecology, 84, 1045-1057.
Wagner, H.H. (2004) Direct multi-scale ordination with canonical correspondence analysis. Ecology, 85, 342-351.

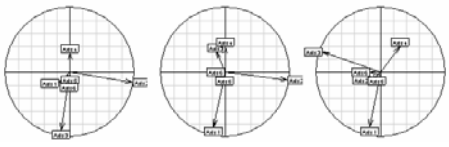
voir également

kmsov
multispati

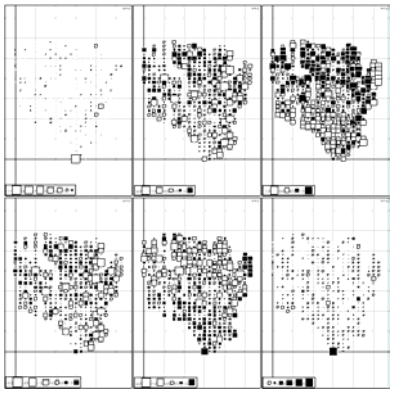
exemples

```
# analyse non symétrique des correspondences à composantes cartographiables
cfi.nsc <- dudi.nsc(CFI$tab)
Select the number of axes: 2
nb <- dnearneigh(as.matrix(CFI$xy), 0, 6)
cfi.msov <- msov(cfi.nsc, nb, scannf = T)
Select the number of axes: 2
summary.msov(cfi.msov)
Multivariate Spatial Analysis
Call: msov(dudi = cfi.nsc, nb = nb, scannf = T)

Scores from the first duality diagramm:
            var      cum     ratio     geary
Axis1 2.8002601 2.800260 0.4068541 0.6076641
Axis2 0.5434165 3.343677 0.4858079 0.6740274

Eigenvalues decomposition:
            eig      cum     ratio       var     geary
Axis1 1.7022817 2.800260 0.4068541 2.7991644 0.6081392
Axis2 0.3665467 3.343677 0.4858079 0.5430409 0.6749892
```

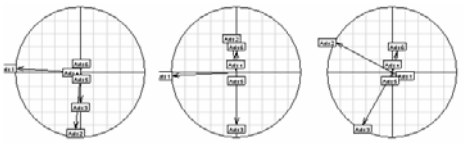
```
cfi.coa <- dudi.coa(CFI$tab)
Select the number of axes: 6
cfi.coa.msov <- msov(cfi.coa, nb)
par(mfrow = c(1, 3))
s.corcircle(cfi.coa.msov$as, 1, 2)
s.corcircle(cfi.coa.msov$as, 1, 3)
s.corcircle(cfi.coa.msov$as, 2, 3)
```



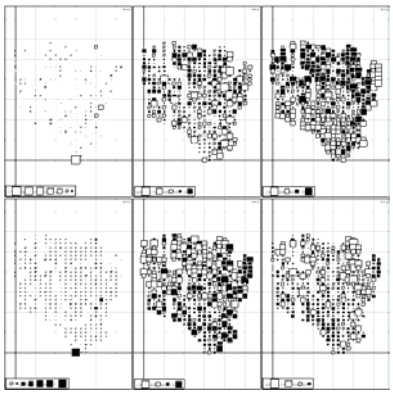
```
par(mfrow = c(2, 3))
s.value(CFI$xy, cfi.coa$li[,1])
s.value(CFI$xy, cfi.coa$li[,2])
s.value(CFI$xy, cfi.coa$li[,3])
s.value(CFI$xy, cfi.coa.msov$li[,1])
s.value(CFI$xy, cfi.coa.msov$li[,2])
s.value(CFI$xy, cfi.coa.msov$li[,3])
```



```
nb <- dnearneigh(as.matrix(CFI$xy), 6, 13)
cfi.coa.msov <- msov(cfi.coa, nb)
par(mfrow = c(1, 3))
s.corcircle(cfi.coa.msov$as, 1, 2)
s.corcircle(cfi.coa.msov$as, 1, 3)
s.corcircle(cfi.coa.msov$as, 2, 3)
```

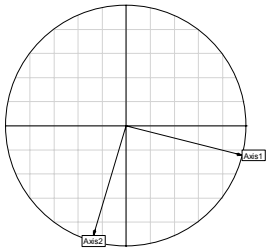


```
par(mfrow = c(2, 3))
s.value(CFI$xy, cfi.coa$li[,1])
s.value(CFI$xy, cfi.coa$li[,2])
s.value(CFI$xy, cfi.coa$li[,3])
s.value(CFI$xy, cfi.coa.msov$li[,1])
s.value(CFI$xy, cfi.coa.msov$li[,2])
s.value(CFI$xy, cfi.coa.msov$li[,3])
```



```
# variations locales
at.xy <- atya$xy
at.gen <- atya$gen
u <- as.character(substitute(donnees))
dirlib <- getwd()
file <- file.path(dirlib, u, "atyacarto.pnm")
at.pnm <- read.pnm(file)
at.nb <- neig2nb(atya$neig)
at.listw <- nb2listw(at.nb)
at.pca <- dudi.pca(at.gen, scale = F)
Select the number of axes: 5
at.msov <- msov(at.pca, at.nb, scannf = TRUE)
Select the number of axes: 5
summary.msov(at.msov)
Multivariate Spatial Analysis
Call: msov(dudi = at.pca, nb = at.nb, scannf = TRUE)

Scores from the first duality diagramm:
             var        cum     ratio      geary
Axis1 0.052108203 0.05210820 0.8133311 0.09277520
Axis2 0.005176584 0.05728479 0.8941298 0.09510982
Axis3 0.002660810 0.05994560 0.9356611 0.07338386
Axis4 0.001515542 0.06146114 0.9593164 0.09946027
Axis5 0.001380038 0.06284118 0.9808568 0.11385019

Eigenvalues decomposition:
               eig        cum     ratio         var      geary
Axis1 0.0048394185 0.05210820 0.8133311 0.052053821 0.09296951
Axis2 0.0004945558 0.05728479 0.8941298 0.005151480 0.09600266
Axis3 0.0002142441 0.05994560 0.9356611 0.002416979 0.08864124
Axis4 0.0001714390 0.06146114 0.9593164 0.001453852 0.11792053
Axis5 0.0001180687 0.06284118 0.9808568 0.001677916 0.07036631

# une information exclusivement spatiale
t3012.nb <- knn2nb(knearneigh(as.matrix(t3012$xy),3))
t3012.neig <- nb2neig(t3012.nb)
t3012.listw <- nb2listw(t3012.nb)
t3012.pca <- dudi.pca(t3012$temp)
Select the number of axes: 2
t3012.pca.msov <- msov(t3012.pca, t3012.nb, scannf = TRUE)
Select the number of axes: 2
summary(t3012.pca.msov)
Multivariate Spatial Analysis
Call: msov(dudi = t3012.pca, nb = t3012.nb, scannf = TRUE)
```

```
Scores from the first duality diagramm:
         var   cum  ratio   geary
Axis1 10.386 10.39 0.8655 0.02708
Axis2  1.508 11.89 0.9912 0.05623

Eigenvalues decomposition:
          eig   cum  ratio    var   geary
Axis1 0.28223 10.39 0.8655 10.349 0.02727
Axis2 0.08551 11.89 0.9912  1.494 0.05722


# croissance et alternance, global et local
clem.neig <- neig(n.line=15)
clem.nb <- neig2nb(clem.neig)
clem.listw <- nb2listw(clem.nb)
clem.pca <- dudi.pca(clementines, scale = FALSE)
Select the number of axes: 2
clem.pca.msov <- msov(clem.pca, clem.nb, scannf = TRUE)
Select the number of axes: 3
s.corcircle(clem.pca.msov$as)
summary(clem.pca.msov)
Multivariate Spatial Analysis
Call: msov(dudi = clem.pca, nb = clem.nb, scannf = TRUE)

Scores from the first duality diagramm:
         var    cum  ratio   geary
Axis1 119761 119761 0.8510 0.08229
Axis2  10059 129821 0.9225 0.19690

Eigenvalues decomposition:
          eig    cum  ratio    var   geary
Axis1 10584.5 119761 0.8510 111261 0.09513
Axis2  1401.5 129821 0.9225  15762 0.08892
Axis3   666.2 133320 0.9474   4158 0.16022
```



R code

```
"msov" <- function(dudi, nb, scannf = FALSE, nf = 6){

require(spdep)
    if (!inherits(dudi, "dudi"))
        stop("object of class 'dudi' expected")
    if (!inherits(nb, "nb"))
        stop("object of class 'nb' expected")

# on construit la matrice de covariance à partir des produits matriciels
X <- as.matrix(dudi$tab)
n <- nrow(X)
p <- ncol(X)
col.w <- dudi$cw
row.w <- dudi$lw
gh <- neig2mat(nb2neig(nb))
mh <- diag(row.w)%*%gh%*%diag(row.w)
i <- as.matrix(rep(1,n))
nh <- diag(as.vector(mh%*%i))
L <- nh-mh
w <- sqrt(col.w)
covh <- t(X)%*%L%*%X
covh <- sweep(covh, 2, w, "*")
covh <- covh * w

# on diagonalise la matrice de covariance et on en déduit les éléments du schéma de
# dualité
res <- list(tab = dudi$tab, cw = col.w, lw = row.w)
eigh <- eigen(covh)
tol <- 1e-07
eig <- eigh$values
rankh <- sum((eig/eig[1]) > tol)
```

```
if (scannf){
barplot(eig[1:rankh])
cat("Select the number of axes: ")
nf <- as.integer(readLines(n = 1))
}
if (nf <= 0)
    nf <- 2
if (nf > rankh)
    nf <- rankh
res$eig <- eig[1:rankh]
res$rankh <- rankh
res$nf <- nf
    # axes principaux
    pond.col <- col.w
    pond.col[which(pond.col == 0)] <- 1
    pond.col <- 1/sqrt(pond.col)
    auxi <- data.frame(eigh$vectors[, 1:nf] * pond.col)
    names(auxi) <- paste("CS", (1:nf), sep = "")
    row.names(auxi) <- names(res$tab)
    res$c1 <- auxi

    # composantes principales
    auxi <- X%*%diag(col.w)%*%as.matrix(res$c1)%*%diag(1/sqrt(eig[1:nf]))
    auxi <- data.frame(auxi)
    names(auxi) <- paste("RS", (1:nf), sep = "")
    row.names(auxi) <- row.names(res$tab)
    res$l1 <- auxi

    # coordonnées des colonnes
    w <- matrix(sqrt(eig[1:nf]), p, nf, byr = TRUE)
    auxi <- data.frame(as.matrix(res$c1) * w)
    names(auxi) <- paste("Comp", (1:nf), sep = "")
    row.names(auxi) <- names(res$tab)
    res$co <- auxi

    # coordonnées des lignes
    w <- matrix(sqrt(eig[1:nf]), n, nf, byr = TRUE)
    auxi <- data.frame(as.matrix(res$l1) * w)
    names(auxi) <- paste("Axis", (1:nf), sep = "")
    row.names(auxi) <- row.names(res$tab)
    res$li <- auxi

    # projection des axes principaux de l'analyse simple sur les axes de l'analyse
    # sous contrainte
    auxi <- as.matrix(res$c1) * unlist(dudi$cw)
    auxi <- data.frame(t(as.matrix(dudi$c1)) %*% auxi)
    row.names(auxi) <- names(dudi$li)
    names(auxi) <- names(res$li)
    res$as <- auxi

res$call <- match.call()
class(res) <- c("msov", "dudi")
return(res)
}

####   ####   ####   ####
####   ####   ####   ####

"summary.msov" <- function(msov, ...){
    # calcule l'inertie des variables d'un tableau pour une pondération w
    norm.w <- function(X, w) {
        f2 <- function(v) sum(v * v * w)/sum(w)
        norm <- apply(X, 2, f2)
        return(norm)
    }

    if (!inherits(msov, "msov"))
        stop("to be used with 'msov' object")
```

```
    cat("\nMultivariate Spatial Analysis\n")
    cat("Call: ")
    print(msov$call)

    appel <- as.list(msov$call)
    dudi <- eval(appel$dudi, sys.frame(0))
    nb <- eval(appel$nb, sys.frame(0))

    # les scores de l'analyse de base
    nf <- dudi$nf
    eig <- dudi$eig[1:nf]
    cum <- cumsum (dudi$eig) [1:nf]
    ratio <- cum/sum(dudi$eig)
    col.w <- dudi$cw
    row.w <- dudi$lw
    gh <- neig2mat(nb2neig(nb))
    mh <- diag(row.w)%*%gh%*%diag(row.w)
    i <- as.matrix(rep(1,length(row.w)))
    nh <- diag(as.vector(mh%*%i))
    L <- nh-mh
    X <- as.matrix(dudi$li)
    geary <- diag(t(X)%*%L%*%X)
    res <- data.frame(var = eig, cum = cum, ratio = ratio, geary = geary/eig)
    cat("\nScores from the first duality diagramm:\n")
    print(res)

    # les scores de l'analyse spatiale
    nf <- msov$nf
    nvar <- nrow(msov$c1)
    eig <- msov$eig[1:nf]
    cum <- cumsum (dudi$eig) [1:nf]
    ratio <- cum/sum(dudi$eig)
    X <- as.matrix(msov$li)
    varspa <- norm.w(X,dudi$lw)
    geary <- diag(t(X)%*%L%*%X)
    res <- data.frame(eig = eig, cum = cum, ratio = ratio, var = varspa, geary =
       geary/varspa)

    cat("\nEigenvalues decomposition:\n")
    print(res)
}
```

12. Ordination sous contrainte spatiale version Moran

alias

multispati
plot.multispati
summary.multispati

description

'multispati' est une fonction qui généralise la mesure de l'autocorrelation d'une variable par l'indice de Moran à l'analyse d'un tableau multivarié. On introduit la contrainte spatiale au sein de l'analyse d'un triplet statistique par l'intermédiaire d'une pondération de voisinage. Cette approche est différente de celle proposée dans Thioulouse et al. (1995) où les auteurs cherchent à réconcilier les deux points de vue Geary-Moran. De plus elle généralise l'approche de Wartenberg sur un lien de type L à l'ensemble des analyses multivariées à un tableau. Elle permet de définir un Moran scatterplot multivarié.
'plot.multispati' est une fonction graphique assurant une représentation graphique canonique des principaux éléments de l'analyse
'summary.multispati' est une fonction donnant un résumé de l'analyse

usage

```
multispati (dudi, listw, scannf = TRUE, nfposi = 2, nfnega = 0)
plot.multispati (x, xax = 1, yax = 2, ...)
summary.multispati (object, ...)
```

arguments

- *dudi* : un objet de la classe 'dudi'. Cet objet est le résultat de l'analyse du triplet statistique $(\mathbf{X},\mathbf{Q},\mathbf{D})$
- *listw* : un objet de la classe 'listw' représentant les pondérations de voisinage associées au graphe de voisinage considéré. L'option 'style' associée à cet objet doit être de la forme 'W' ce qui indique que la matrice poids de voisinage a été préalablement normalisée par lignes
- *scannf* : une variable binaire. Si TRUE, le nombre de valeurs propres à retenir est demandé à l'utilisateur. Sinon, ce nombre égale les arguments *nfposi* et *nfnega*
- *nfposi* : un entier désignant le nombre de valeurs propres positives à conserver si *scannf* = FALSE
- *nfnega* : un entier désignant le nombre de valeurs propres négatives à conserver si *scannf* = FALSE
- *x, object* : un objet de la classe 'multispati'
- *xax* : le numéro de la colonne correspondant à l'axe des x
- *yax* : le numéro de la colonne correspondant à l'axe des y
- … : arguments supplémentaires

détails

On note $(\mathbf{X}_C,\mathbf{Q},\mathbf{D})$ le triplet de l'analyse statistique considérée. L'analyse de base passe par la décomposition en valeur singulière (svd) du tableau $\mathbf{Q}^{1/2}\mathbf{X}_C^t\mathbf{D}^{1/2}$, ou par l'analyse (eigenanalysis) de la matrice $\mathbf{Q}^{1/2}\mathbf{X}_C^t\mathbf{D}\mathbf{X}_C\mathbf{Q}^{1/2}$. Les valeurs propres obtenues maximise l'inertie projetée et assure une décomposition optimale de l'inertie du tableau : $tr\left(\mathbf{Q}^{1/2}\mathbf{X}_C^t\mathbf{D}\mathbf{X}_C\mathbf{Q}^{1/2}\right)=\sum_{i=1}^{r}\lambda_i$.

On note $\mathbf{L}$ l'opérateur de retard qui correspond à une matrice de pondération de voisinage normalisée par ligne. Il permet de calculer $\mathbf{LX}$, la moyenne des valeurs par voisins de chaque point pour chaque variable. Pour une variable $\mathbf{x}$ on a $\mathbf{y}=\mathbf{Lx}$ et la représentation des couples $(\mathbf{x},\mathbf{y})$ donne une image canonique de l'autocorrelation de $\mathbf{x}$ appelée *scatterplot* d'Anselin. Les axes principaux de l'analyse simple maximise la variance projetée et n'ont aucune propriété quant à leur structure spatiale. On peut partir du coefficient de Moran pour définir un nouveau critère à maximiser : $I(\mathbf{XQv})=\frac{\mathbf{v}^t\mathbf{QX}^t\mathbf{DLXQv}}{\mathbf{v}^t\mathbf{QX}^t\mathbf{DXQv}}$. Considérons alors la matrice $\mathbf{H}=\frac{1}{2}\mathbf{X}^t\left(\mathbf{L}^t\mathbf{D}+\mathbf{DL}\right)\mathbf{XQ}$.
Elle est $\mathbf{Q}$-symétrique et possède une base de vecteurs propres $\mathbf{Q}$-orthonormés. Le premier vecteur propre $\mathbf{v}_1$ associé à la plus grande valeur propre λ_1 maximise sous

contrainte d'orthonormalité $\|\mathbf{v}\|_{\mathbf{Q}} = 1$ la quantité $\langle \mathbf{Hv} | \mathbf{v} \rangle_{\mathbf{Q}} = \mathbf{v}'\mathbf{QX}'\mathbf{DLXQv} = \|\mathbf{XQv}\|_{\mathbf{D}}^2 I(\mathbf{XQv})$. On maximise donc un compromis entre l'inertie projetée et la mesure de l'autocorrélation donnée par l'indice de Moran. La carte factorielle obtenue garde en partie la propriété de maximiser l'inertie projetée mais intègre également le lien de voisinage. Elle donne un bon compromis entre la carte factorielle classique et le scatterplot d'Anselin. Cette image canonique justifie à elle seule l'existence de l'analyse. Pour une ACP normée, on retrouve l'analyse de Wartenberg (1985b) pour une pondération de voisinage normalisée par ligne.

La fonction 'summary.multispati' donne d'une part un résumé des statistiques d'inertie associées à l'analyse simple. La colonne 'eig' correspond aux valeurs propres, qui donnent l'inertie projetée, c'est-à-dire la variance des coordonnées sur les axes principaux $\|\mathbf{XQv}\|_{\mathbf{D}}^2$. La colonne 'cum' donne la somme cummulée. La colonne ratio donne le pourcentage d'inertie expliquée. La colonne 'moran' donne le l'indice de Moran des coordonnées sur chaque axe $I(\mathbf{XQv})$. Elle donne ensuite les statistiques associées à l'analyse sous contrainte. La colonne 'eig' donne les valeurs propres. Cette fois-ci les valeurs propres correspondent à l'autocovariance des coordonnées sur chaque axe $\mathbf{v}'\mathbf{QX}'\mathbf{DLXQv} = I(\mathbf{XQv})\|\mathbf{XQv}\|_{\mathbf{D}}^2$. Les colonnes 'cum' est 'ratio' sont définies comme précédemment. La colonne 'var' donne la variance des coordonnées sur chaque axe conservé $\|\mathbf{XQv}\|_{\mathbf{D}}^2$. La colonne 'moran' donne l'indice de Moran des coordonnées $I(\mathbf{XQv})$.

valeurs

'multispati' retourne un objet de la classe 'multispati' qui contrairement à 'msov' n'est pas une sous classe de la classe 'dudi'. L'analyse sous contrainte peut être définie comme l'analyse du quadruplet $(\mathbf{X}_C, \mathbf{Q}, \mathbf{D}, \mathbf{L})$.

références

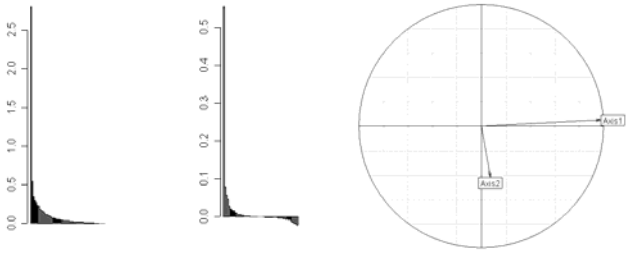
Ollier, S., Dray, S., & Chessel, D. (soumis) Taking into account spatial dependence in multivariate analysis: a generalization of Wartenberg's multivariate spatial correlation. Geographical Analysis.
Wartenberg, D.E. (1985) Multivariate spatial correlations: a method for exploratory geographical analysis. Geographical Analysis, 17, 263-283.

voir également

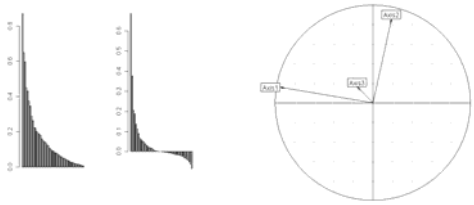
multispati.randtest
msov

exemples

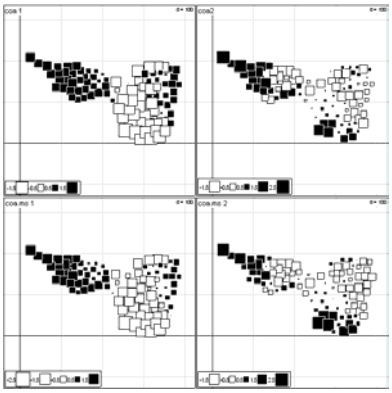
```
# analyse non symétrique des correspondances à composantes cartographiables
par(mfrow = c(1,2))
cfi.nsc <- dudi.nsc(CFI$tab)
Select the number of axes: 2
CFI.nb <- dnearneigh(as.matrix(CFI$xy), 0, 0.8)
CFI.listw <- nb2listw(CFI.nb)
cfi.nsc.ms <- multispati(cfi.nsc, CFI.listw)
Select the first number of axes (>=1): 4
Select the second number of axes (>=0): 0
s.corcircle(cfi.nsc.ms$as)

summary(cfi.nsc.ms)
Multivariate Spatial Analysis
Call: multispati(dudi = cfi.nsc, listw = CFI.listw)

Scores from the first duality diagramm:
       var   cum  ratio   moran
RS1 2.8003 2.800 0.4069 0.18936
RS2 0.5434 3.344 0.4858 0.05549

Eigenvalues decomposition:
        eig    var  moran
CS1 0.55316 2.6912 0.2055
CS2 0.07959 0.2648 0.3006
CS3 0.05700 0.2619 0.2176
CS4 0.04635 0.2265 0.2046

# analyse des correspondances à composantes cartographiables
```



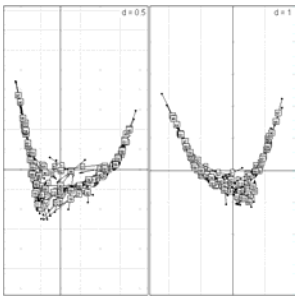
```
xy <- mafragh$xy
flo <- mafragh$flo
flo <- mafragh$flo[,apply(mafragh$flo,2,sum)>3]
flonames <- mafragh$espnames[apply(mafragh$flo,2,sum)>3]
# on conserve 97 sites et 49 espèces dont l'abondance vaut plus de 3.
maf.listw <- nb2listw(neig2nb(mafragh$neig))
par(mfrow = c(1,2))
maf.coa <- dudi.coa(flo)
Select the number of axes: 3
maf.coa.ms <- multispati(maf.coa,maf.listw)
Select the first number of axes (>=1): 2
Select the second number of axes (>=0): 0
s.corcircle(maf.coa.ms$as)
```



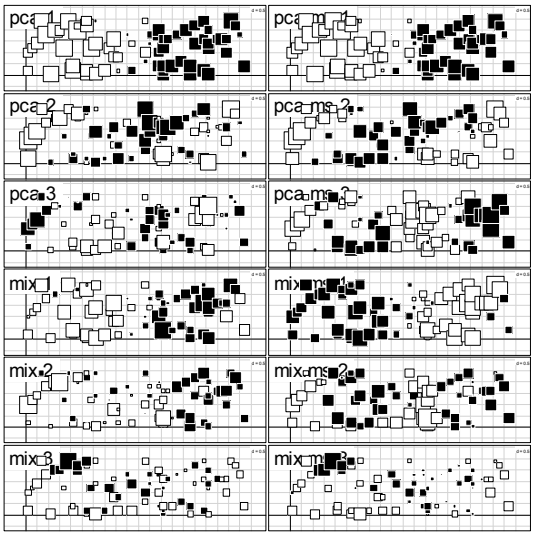
```
par(mfrow = c(2, 2))
s.value(xy,-maf.coa$li[,1],
 sub = "coa 1")
s.value(xy,maf.coa$li[,2],
 sub = "coa2")
s.value(xy,maf.coa.ms$li[,1],
 sub = "coa.ms 1")
s.value(xy,maf.coa.ms$li[,2],
 sub = "coa.ms 2")
```



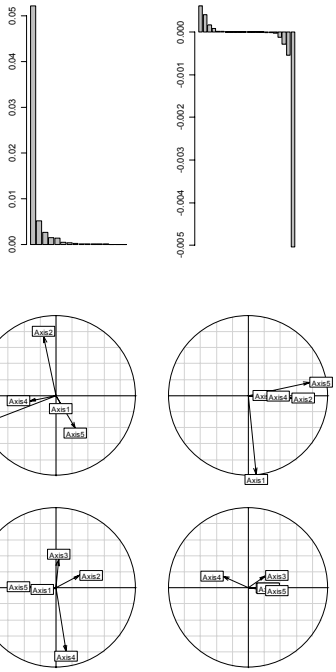
```
par(mfrow = c(1, 2))
w1 <- maf.coa$li[,1:2]
w1m <- apply(w1,2,function(
 var) lag.listw (x = maf.listw, var))
s.match(w1, w1m, clab = 0.5)
w1 <- maf.coa.ms$li[,1:2]
w1m <- apply(w1,2, function(
 var)  lag.listw (x = maf.listw, var))
s.match(w1, w1m, clab = 0.50)
```

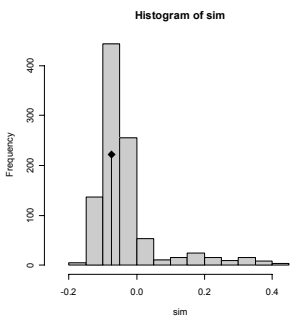


```
# ACP et analyse mixte à composantes cartographiables
ori.xy <- oribatid$xy[,c(2,1)]
names(ori.xy) <- c("x","y")
ori.nb <- knn2nb(knearneigh(as.matrix(ori.xy), 3))
ori.listw <- nb2listw(ori.nb)
ori.log <- log(oribatid$fau+1)
ori.pca <- dudi.pca(ori.log,scale=F)
Select the number of axes: 4
ori.mix <- dudi.mix(oribatid$envir)
Select the number of axes: 3
ori.pca.ms <- multispati(ori.pca, ori.listw)
Select the first number of axes (>=1): 3
Select the second number of axes (>=0): 0
ori.mix.ms <- multispati(ori.mix, ori.listw)
Select the first number of axes (>=1): 3
Select the second number of axes (>=0): 0
par(mfcol=c(6,2))
 for (i in 1:3) s.value(ori.xy,
                    ori.pca$l1[,i],
                    sub=paste("pca",i),
                    csub=3,cleg=0)
 for (i in 1:3) s.value(ori.xy,
                    ori.mix$l1[,i],
                    sub=paste("mix",i),
                    csub=3,cleg=0)
 for (i in 1:3) s.value(ori.xy,
                     ori.pca.ms$li[,i],
                     sub=paste("pca.ms",i),
```



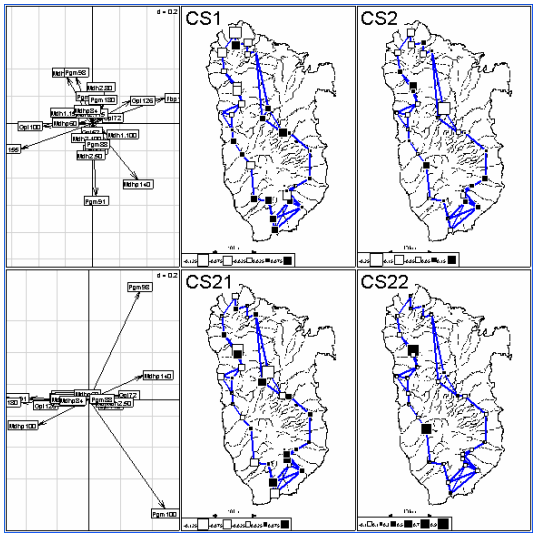
```
                    csub=3, cleg=0)
 for (i in 1:3) s.value(ori.xy,
                    ori.mix.ms$li[,i],
                    sub=paste("mix.ms",i),
                    csub=3, cleg=0)


# l'information est très spatialisée: les gains sont minimes avec les analyses sous
# contraintes
# il y a plusieurs échelles de structuration spatiale, d'où l'article de Borcard
# (2004)
# le lien avec l'espace est une contrainte qui peut cacher des relations non
# spatialisées entre faune et environnement d'où les méthodes visant à se
# débarrasser des contraintes (Borcard et al., 1992)

# variations locales
at.xy <- atya$xy
at.gen <- atya$gen
u <- as.character(substitute(donnees))
dirlib <- getwd()
file <- file.path(dirlib, u, "atyacarto.pnm")
at.pnm <- read.pnm(file)
at.nb <- neig2nb(atya$neig)
at.listw <- nb2listw(at.nb)
par(mfrow = c(1, 2))
at.pca <- dudi.pca(at.gen, scale = F)
Select the number of axes: 5
at.pca.ms <- multispati(at.pca, at.listw)
Select the first number of axes (>=1): 4
Select the second number of axes (>=0): 4

par(mfrow = c(2,2))
s.corcircle(at.pca.ms$as)
s.corcircle(at.pca.ms$as, 7, 8)
s.corcircle(at.pca.ms$as, 5, 6)
s.corcircle(at.pca.ms$as, 3, 4)
summary(at.pca.ms)

Multivariate Spatial Analysis
Call: multispati(dudi = at.pca, listw = at.listw)

Scores from the first duality diagramm:
         var     cum  ratio     moran
RS1 0.052108 0.05211 0.8133 -0.092958
RS2 0.005177 0.05728 0.8941  0.009872
RS3 0.002661 0.05995 0.9357  0.155230
RS4 0.001516 0.06146 0.9593 -0.085363
RS5 0.001380 0.06284 0.9809 -0.263020

Eigenvalues decomposition:
            eig       var   moran
CS1    6.051e-04 0.0023710  0.2552
CS2    4.048e-04 0.0038902  0.1041
CS3    1.580e-04 0.0011283  0.1400
CS4    8.462e-05 0.0004893  0.1729
CS19 -1.213e-04 0.0008058 -0.1505
CS20 -2.918e-04 0.0014903 -0.1958
CS21 -5.488e-04 0.0029479 -0.1862
CS22 -5.031e-03 0.0502431 -0.1001

gearymoran(bilis = listw2mat(at.listw), at.pca.ms$li)
class: krandtest
test number:   8
permutation number:   999
  test obs    P(X<=obs) P(X>=obs)
1 CS1  0.257  0.974     0.028
2 CS2  0.105  0.849     0.153
3 CS3  0.142  0.876     0.126
```





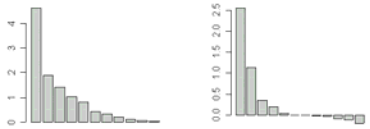
```
4 CS4  0.175  0.912      0.09
5 CS19 -0.15  0.2        0.802
6 CS20 -0.194 0.16       0.842
7 CS21 -0.188 0.125      0.877
8 CS22 -0.102 0.21       0.792
```

```
at.testglobal <- multispati.randtest(at.pca, at.listw)
plot(at.testglobal)
```



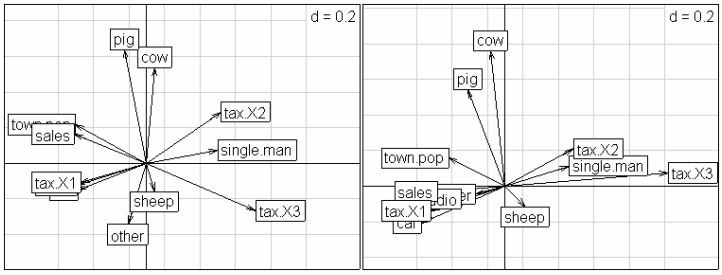

```
par(mfrow = c(2, 3))
par(mar = c(0,0,0,0))
s.arrow(at.pca.ms$c1, 1, 2)
for(i in 1:2){
plot(at.pnm)
plot(at.nb, at.xy,col="blue",add=T,lwd=2)
s.value(at.xy, at.pca.ms$li[,i], add.p=T,
 csi=1, sub = names(at.pca.ms$li)[i],
 csub = 3)
}
s.arrow(at.pca.ms$c1, 7, 8)
for(i in 7:8){
plot(at.pnm)
plot(at.nb, at.xy,col="blue",add=T,lwd=2)
s.value(at.xy, at.pca.ms$li[,i], add.p=T,
 csi=1, sub = names(at.pca.ms$li)[i],
 csub = 3)
}
```

```
# cartes factorielles et cartes spatiales
# on regroupe les trois premières variables en 1 facteur à trois modalités
# on passe d'un tableau de variables quantitatives à un tableau mixte
w <- irishdata$tab[,1:3]
tax <- as.factor(kmeans(w,w[c(1,3,5),])$cluster)
tab <- cbind.data.frame(irishdata$tab[,-(1:3)],tax)
ana1 <- dudi.mix(tab)
Select the number of axes: 4

# on construit la matrice des pondérations de voisinages à partir de la longueur
# des fontières
w <- irishdata$link.utm
w1 <- t(apply(w,1,function(x) x/sum(x)))
apply(w1,1,sum)
   Carlow     Cavan     Clare      Cork   Donegal    Galway    Kerry   Kildare
        1         1         1         1         1         1         1         1
 Kilkenny   Laoghis   Leitrim  Limerick  Longford     Louth     Mayo     Meath
        1         1         1         1         1         1         1         1
 Monaghan    Offaly Roscommon     Sligo Tipperary Waterford Westmeath   Wexford
        1         1         1         1         1         1         1         1
  Wicklow
        1
w2 <- mat2listw(w1)
w2$style <- "W"
ms1 <- multispati(ana1,w2)
Select the first number of axes (>=1): 4
Select the second number of axes (>=0): 0
```

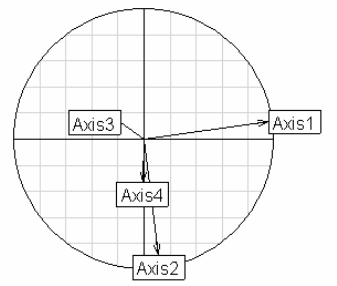


```
par(mfrow=c(1,2))
barplot(ana1$eig,col=grey(0.8))
barplot(ms1$eig,col=grey(0.8))
```

```
ana1$c1[,2]=- ana1$c1[,2]
ana1$co[,2]=- ana1$co[,2]
ana1$li[,2]=- ana1$li[,2]
ana1$l1[,2]=- ana1$l1[,2]
s.arrow(ana1$c1,xlim=c(-0.8,1.2))
s.arrow(ms1$c1,xlim=c(-0.8,1.2))
```

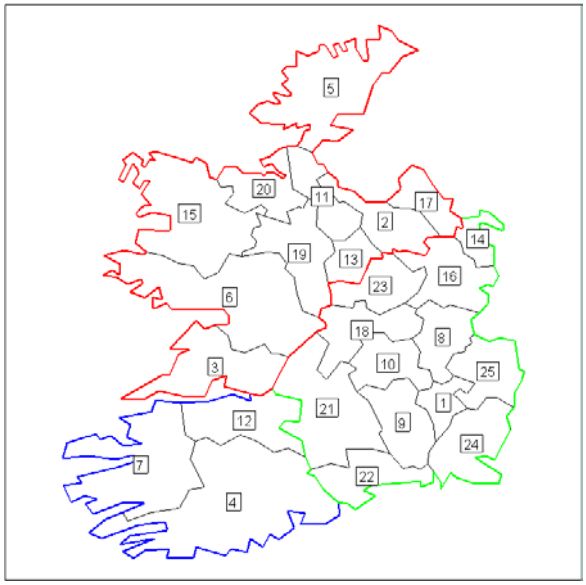


```
s.corcircle(ms1$as)
```



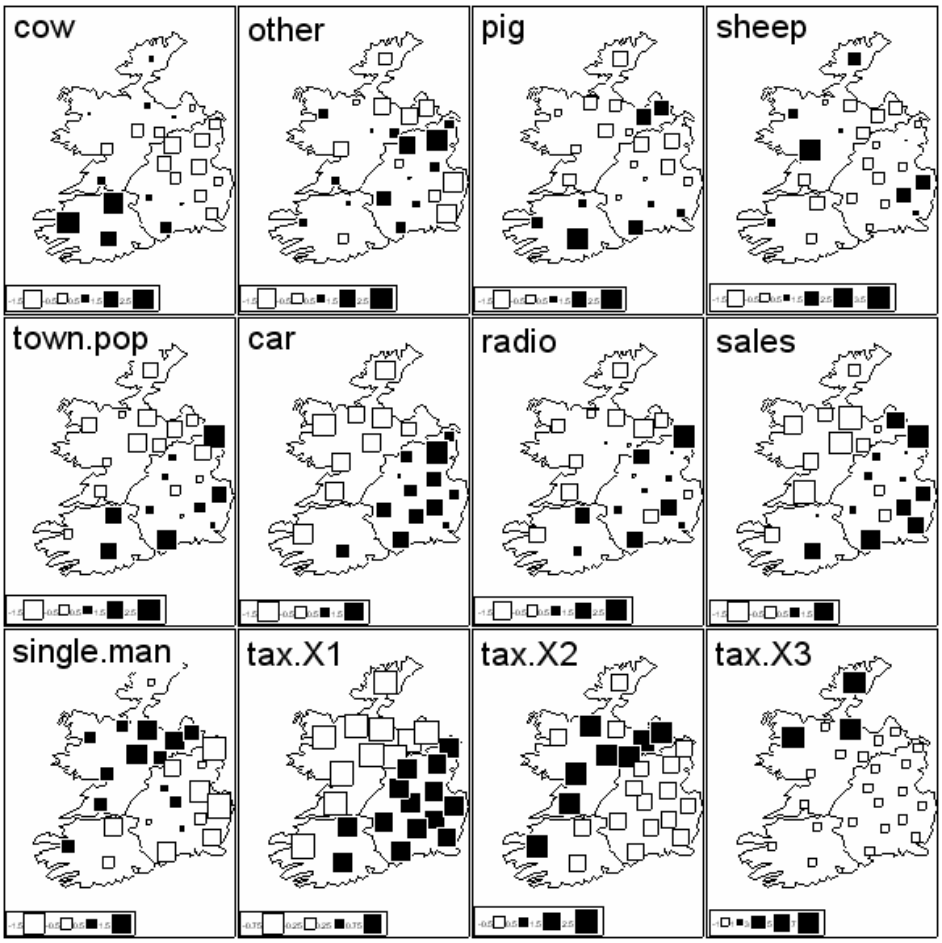

```
ana1$ls=apply(ana1$li,2,function(
 var) lag.listw(w2,var))
s.match(ana1$li,ana1$ls)
s.match(ms1$li,ms1$ls)
```

```
z <- area2poly(irishdata$area.utm)
z1 <- lapply(c(4,7,12),function(k) z[[k]])
z2 <- lapply(c(1,8:10,14,16,18,21:25),
 function(k) z[[k]])
z3 <- lapply(c(2:3,5:6,11,13,15,17,19,20),
 function(k) z[[k]])
class(z1) <- "polylist"
class(z2) <- "polylist"
class(z3) <- "polylist"
area.plot(irishdata$area.utm)
f1 <- function(x) segments(x[1], x[2],
 x[3], x[4], col="blue", lwd=2)
 apply(area.util.contour(poly2area(z1)),1,f1)
f1 <- function(x) segments(x[1], x[2],
 x[3], x[4], col="green", lwd=2)
 apply(area.util.contour(poly2area(z2)),1,f1)
f1 <- function(x) segments(x[1], x[2],
 x[3], x[4], col="red", lwd=2)
 apply(area.util.contour(poly2area(z3)),1,f1)
s.label(irishdata$xy.utm,
 label=as.character(1:25),
 add.p=T)
```

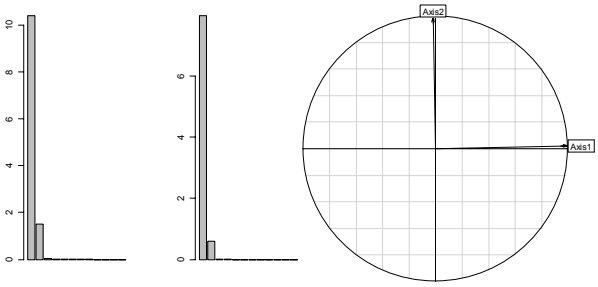


```
cont1 <- rbind.data.frame(area.util.contour(
 poly2area(z1)),area.util.contour(poly2area(z2)),
 area.util.contour(poly2area(z3)))
par(mfrow = c(1,2))
xl <- c(-50,300)
yl <- c(5750,6100)
s.value(irishdata$xy.utm,ms1$li[,1],
 contour=cont1,xlim=xl,ylim=yl,
 adda=F,grid=F,csi=1.5)
s.value(irishdata$xy.utm,ms1$li[,2],
 contour=cont1,xlim=xl,ylim=yl,
 adda=F,grid=F,csi=1.5)


par(mfrow = c(3,4))
xl <- c(-50,300)
```

```
yl <- c(5750,6100)
for (k in 1:12) {
s.value(irishdata$xy.utm,ana1$tab[,k],
 contour=cont1,xlim=xl,ylim=yl,adda=F,
 grid=F,csi=1.5,sub=names(ana1$tab)[k],csub=3)}
```

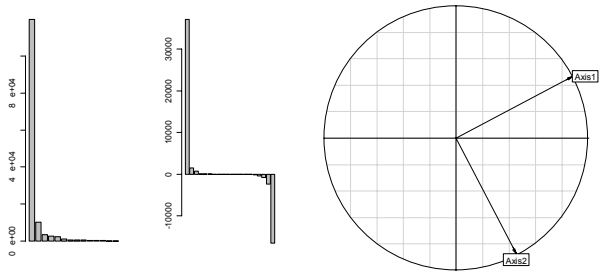



```
# une information exclusivement spatiale
t3012.nb <- knn2nb(knearneigh(
 as.matrix(t3012$xy),3))
t3012.neig <- nb2neig(t3012.nb)
t3012.listw <- nb2listw(t3012.nb)
par(mfrow = c(1,2))
t3012.pca <- dudi.pca(t3012$temp)
Select the number of axes: 2
t3012.pca.ms <- multispati(
 t3012.pca, t3012.listw)
Select the first number of axes (>=1): 2
Select the second number of axes (>=0): 0
s.corcircle(t3012.pca.ms$as)

summary(t3012.pca.ms)
```

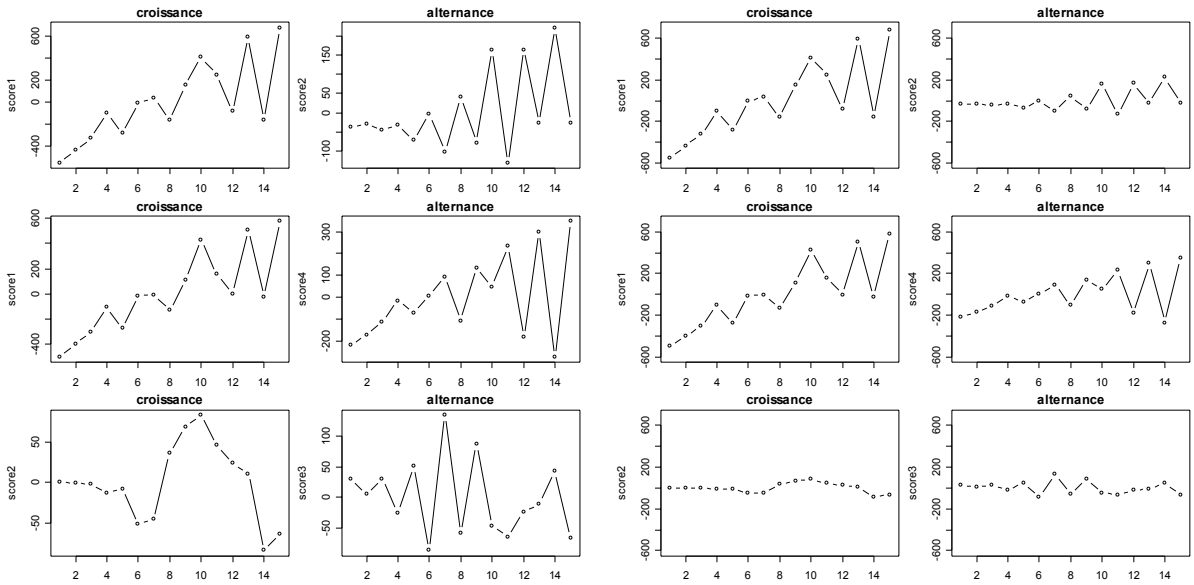



```
Multivariate Spatial Analysis
Call: multispati(dudi = t3012.pca, listw = t3012.listw)

Scores from the first duality diagramm:
       var   cum  ratio  moran
RS1 10.386 10.39 0.8655 0.7663
RS2  1.508 11.89 0.9912 0.3942

Eigenvalues decomposition:
       eig    var  moran
```

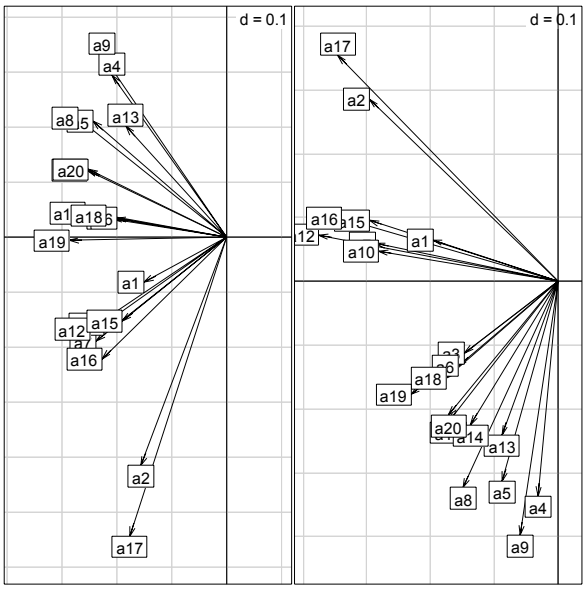
```
CS1 7.9646 10.379 0.7674
CS2 0.6013  1.487 0.4044
# le gain d'une analyse sous contrainte est quasi nul

# croissance et alternance, global et local
clem.neig <- neig(n.line=15)
clem.nb <- neig2nb(clem.neig)
clem.listw <- nb2listw(clem.nb)
par(mfrow = c(1, 2))
clem.pca <- dudi.pca(clementines, scale = FALSE)
Select the number of axes: 2
clem.pca.ms <- multispati(clem.pca, clem.listw)
Select the first number of axes (>=1): 2
Select the second number of axes (>=0): 2
s.corcircle(clem.pca.ms$as, 1, 4)
```

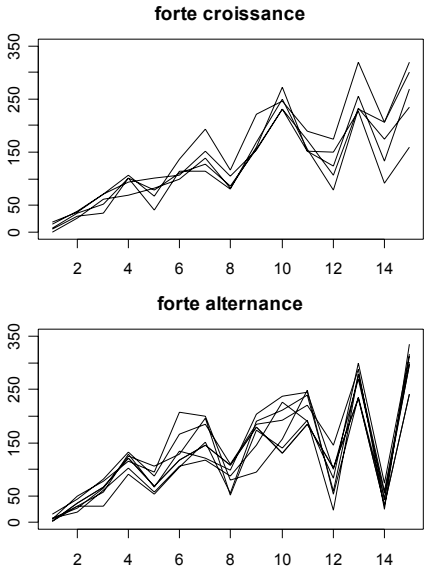



```
summary(clem.pca.ms)

Multivariate Spatial Analysis
Call: multispati(dudi = clem.pca, listw = clem.listw)

Scores from the first duality diagramm:
       var    cum  ratio   moran
RS1 119761 119761 0.8510  0.2096
RS2  10059 129821 0.9225 -0.5008

Eigenvalues decomposition:
        eig   var   moran
CS1   37104 94738  0.3916
CS2    1475  2157  0.6838
CS19  -2285  3637 -0.6284
CS20 -16496 33789 -0.4882

par(mfrow = c(3,2))
par(mar = c(2,4,2,0.5))
plot(-clem.pca$li[,1], type = "b", ylab = "score1", xlab = "", main = "croissance")
plot(-clem.pca$li[,2], type = "b", ylab = "score2", xlab = "", main = "alternance")
plot(-clem.pca.ms$li[,1], type = "b", ylab = "score1", xlab = "", main =
"croissance")
plot(-clem.pca.ms$li[,4], type = "b", ylab = "score4", xlab = "", main =
"alternance")
plot(clem.pca.ms$li[,2], type = "b", ylab = "score2", xlab = "", main =
"croissance")
plot(clem.pca.ms$li[,3], type = "b", ylab = "score3", xlab = "", main =
"alternance")

par(mfrow = c(3,2))
par(mar = c(2,4,2,0.5))
plot(-clem.pca$li[,1], type = "b", ylab = "score1", xlab = "", main = "croissance",
ylim = c(-600, 700))
plot(-clem.pca$li[,2], type = "b", ylab = "score2", xlab = "", main = "alternance",
ylim= c(-600, 700))
plot(-clem.pca.ms$li[,1], type = "b", ylab = "score1", xlab = "", main =
"croissance", ylim = c(-600, 700))
plot(-clem.pca.ms$li[,4], type = "b", ylab = "score4", xlab = "", main =
"alternance", ylim = c(-600, 700))
plot(clem.pca.ms$li[,2], type = "b", ylab = "score2", xlab = "", main =
"croissance", ylim = c(-600, 700))
plot(clem.pca.ms$li[,3], type = "b", ylab = "score3", xlab = "", main =
"alternance", ylim = c(-600, 700))
```

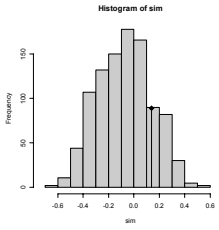



```
par(mfrow = c(1,2))
s.arrow(clem.pca$c1)
s.arrow(clem.pca.ms$c1,1,4)
```






```
par(mfrow = c(2, 1))
par(mar = c(2,2,3,0.5))
ts.plot(clementines[,c(1,10,12,15,16)], ylim = c(0, 350), main = "forte
croissance", xlab = "")
ts.plot(clementines[,c(4,9,5,13,8,11,20,13)], ylim = c(0, 350), main = "forte
alternance", xlab = "")

gearymoran(listw2mat(clem.listw), clem.pca$li)
class: krandtest
test number:   2
permutation number:   999
  test   obs     P(X<=obs) P(X>=obs)
1 Axis1 0.226  0.896     0.106
2 Axis2 -0.466 0.032     0.97

gearymoran(listw2mat(clem.listw), clem.pca.ms$li)
class: krandtest
test number:   4
permutation number:   999
```



```
  test obs    P(X<=obs) P(X>=obs)
1 CS1  0.423  0.972     0.03
2 CS2  0.668  1         0.001
3 CS19 -0.639 0.01      0.992
4 CS20 -0.508 0.034     0.968
```

```
clem.testglobal <- multispati.randtest(clem.pca, clem.listw)
plot(clem.testglobal)
```

R code

```
"multispati" <- function(dudi, listw, scannf = TRUE, nfposi = 2, nfnega = 0) {
    if(!inherits(dudi,"dudi")) stop ("object of class 'dudi' expected")
    if(!inherits(listw,"listw")) stop ("object of class 'listw' expected")
    if(listw$style!="W") stop ("object of class 'listw' with style 'W' expected")
    nvar <- ncol (dudi$tab)
    dudi$cw <- dudi$cw
    row.w <- dudi$lw
    fun <- function (x) lag.listw(listw, x, TRUE)
    tablag <- apply(dudi$tab, 2, fun)
    covar <- t(tablag)%*%as.matrix((dudi$tab*dudi$lw))
    covar <- (covar+t(covar))/2
    covar <- covar * sqrt(dudi$cw)
    covar <- t(t(covar) * sqrt(dudi$cw))
    covar <- eigen(covar, sym = TRUE)
    if (scannf) {
        barplot(covar$values)
        cat("Select the first number of axes (>=1): ")
        nfposi <- as.integer(readLines(n = 1) )
        cat("Select the second number of axes (>=0): ")
        nfnega <- as.integer(readLines(n = 1))
    }
    if (nfposi <= 0)  nfposi <- 1
    if (nfnega <= 0) nfnega = 0
    res <- list()
    res$eig <- covar$values
    res$nfposi <- nfposi
    res$nfnega <- nfnega
    agarder = c(1:nfposi,if (nfnega>0) (nvar-nfnega+1):nvar else NULL)
    agarder = unique (agarder)
    agarder = agarder[which(agarder<=nvar)]
    agarder = agarder[which(agarder>=1)]
    dudi$cw[which(dudi$cw == 0)] <- 1

    # axes principaux
    auxi <- data.frame(covar$vectors[, agarder] /sqrt(dudi$cw))
    names(auxi) <- paste("CS", agarder, sep = "")
    row.names(auxi) <- names(dudi$tab)
    res$c1 <- auxi

    # coordonnées des lignes
    auxi <- as.matrix(auxi)*dudi$cw
    auxi1 <- as.matrix(dudi$tab)%*%auxi
    auxi1 <- data.frame(auxi1)
    names(auxi1) <- names(res$c1)
    row.names(auxi1) <- row.names(dudi$tab)
    res$li <- auxi1

    # lag.vector des coordonnées des lignes
    auxi1 <- as.matrix(tablag)%*%auxi
    auxi1 <- data.frame(auxi1)
    names(auxi1) <- names(res$c1)
    row.names(auxi1) <- row.names(dudi$tab)
    res$ls <- auxi1

    # projection des axes principaux de l'analyse simple sur ceux de l'analyse sous
    # contrainte
    auxi <- as.matrix(res$c1) * unlist(dudi$cw)
    auxi <- data.frame(t(as.matrix(dudi$c1)) %*% auxi)
    row.names(auxi) <- names(dudi$li)
```

```
    names(auxi) <- names(res$li)
    res$as <- auxi


    res$call <- match.call()
    class(res) <- "multispati"
    return(res)
}

"plot.multispati" <- function (x, xax = 1, yax = 2, ...) {
    if (!inherits(x, "multispati"))
        stop("Use only with 'multispati' objects")

    appel <- as.list(x$call)
    dudi <- eval(appel$dudi, sys.frame(0))
    listw <- eval(appel$listw, sys.frame(0))
    nf <- x$nfposi + x$nfnega
    if ((nf == 1) || (xax == yax)) {
        sco.quant(x$li[, 1], dudi$tab)
        return(invisible())
    }
    if (xax > nf)
        stop("Non convenient xax")
    if (yax > nf)
        stop("Non convenient yax")
   f1 <- function ()
    {
        opar <- par(mar = par("mar"))
        on.exit(par(opar))
        m <- length(x$eig)
        par(mar = c(0.8, 2.8, 0.8, 0.8))
        col.w <- rep (grey(1), m)
        col.w[1:x$nfposi] = grey(0.8)
        if (x$nfnega>0) col.w[m:(m-x$nfnega+1)] = grey(0.8)
        j1 = xax
        if (j1>x$nfposi) j1 = j1-x$nfposi +m -x$nfnega
        j2 = yax
        if (j2>x$nfposi) j2 = j2-x$nfposi +m -x$nfnega
        col.w[c(j1,j2)] = grey(0)
         barplot(x$eig, col = col.w)
        scatterutil.sub(cha ="Eigen values", csub = 2, possub = "topright")
    }

    # sortie graphique
    def.par <- par(no.readonly = TRUE)
    on.exit(par(def.par))
    nf <- layout(matrix(c(3, 3, 1, 3, 3, 2), 3, 2))
    par(mar = c(0.2, 0.2, 0.2, 0.2))
       # diagramme des valeurs propres
    f1()
       # axes principaux: représentation des poids canoniques
    s.arrow(x$c1, xax = xax, yax = yax, sub = "Canonical weights", csub = 2, clab =
1.25)
       # coordonnées des lignes/coordonnées des moyennes par voisins: Moran
       # scatterplot bivarié
    s.match(x$li, x$ls, xax = xax, yax = yax, sub = "Classes", csub = 2, clab =
0.75)
}

"summary.multispati" <- function (object, ...) {
    util <- function(n) {
        x <- "1"
        for (i in 2:n) x[i] <- paste(x[i - 1], i, sep = "+")
        return(x)
    }
    norm.w <- function(X, w) {
        f2 <- function(v) sum(v * v * w)/sum(w)
        norm <- apply(X, 2, f2)
```

```
        return(norm)
    }

    if (!inherits(object, "multispati"))
        stop("to be used with 'multispati' object")
    cat("\nMultivariate Spatial Analysis\n")
    cat("Call: ")
    print(object$call)

    appel <- as.list(object$call)
    dudi <- eval(appel$dudi, sys.frame(0))
    listw <- eval(appel$listw, sys.frame(0))

    # les scores de l'analyse de base
    nf <- dudi$nf
    eig <- dudi$eig[1:nf]
    cum <- cumsum (dudi$eig) [1:nf]
    ratio <- cum/sum(dudi$eig)
    w <- apply(dudi$l1, 2, lag.listw, x = listw)
    moran <- apply(w*as.matrix(dudi$l1)*dudi$lw, 2, sum)
    res <- data.frame(var = eig, cum = cum, ratio = ratio, moran = moran)
    cat("\nScores from the first duality diagramm:\n")
    print(res)

    # les scores de l'analyse spatiale
    nfposi <- object$nfposi
    nfnega <- object$nfnega
    nvar <- nrow(object$c1)
    nf <- nfposi + nfnega
    agarder <- c(1:nfposi,if (nfnega>0) (nvar-nfnega+1):nvar else NULL)
    eig <- object$eig[agarder]
    varspa <- norm.w(object$li,dudi$lw)
    moran <- apply(as.matrix(object$li)*as.matrix(object$ls)*dudi$lw,2,sum)
    res <- data.frame(eig = eig, var = varspa, moran = moran/varspa)

    cat("\nEigenvalues decomposition:\n")
    print(res)
}
```

13. Test de l'autocorrelation spatiale multivariée

alias

multispati.randtest
multispati.rtest

description

'multispati.randtest' et 'multispati.rtest' sont des fonctions qui mettent en œuvre un test de permutation multivarié basé sur la mesure de l'autocorrelation spatiale d'un tableau définie par Smouse et Peakall (1999). 'multispati.rtest' fait le calcul exclusivement dans R et 'multispati.randtest' le fait en C avec appel d'une fonction externe.

usage

```
multispati.randtest(dudi, listw, nrepet = 999)
multispati.rtest(dudi, listw, nrepet = 99)
```

arguments

- *dudi* : un objet de la classe 'dudi'. Cet objet est le résultat de l'analyse du triplet statistique $(\mathbf{X},\mathbf{Q},\mathbf{D})$
- *listw* : un objet de la classe 'listw' représentant les pondérations de voisinage associées au graphe de voisinage considéré. L'option 'style' associée à cet objet doit être de la forme 'W' ce qui indique que la matrice poids de voisinage a été préalablement normalisée par lignes
- *nrepet* : un entier correspondant au nombre de permutations

details

On note $(\mathbf{X}_C,\mathbf{Q},\mathbf{D})$ le triplet de l'analyse statistique considérée. On note $\mathbf{L}$ l'opérateur de retard qui correspond à une matrice de pondération de voisinage normalisée par ligne. Le test est basé sur la corrélation de Smouse et Peakall (1999) qui s'écrit dans le cas général : $r = \frac{tr\left(\mathbf{X}_C^t\mathbf{DLX}_C\mathbf{Q}\right)}{tr\left(\mathbf{X}_C^t\mathbf{DX}_C\mathbf{Q}\right)}$.

valeurs

'multispati.randtest' et multispati.rtest' retournent un objet de la classe 'rtest'.

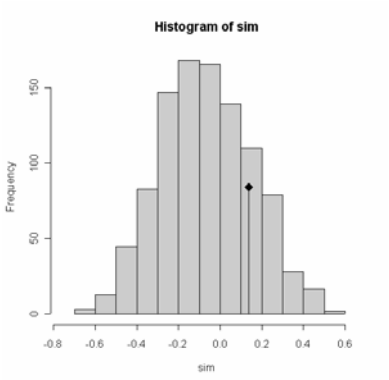
références

Ollier, S., Dray, S., & Chessel, D. (soumis) Taking into account spatial dependence in multivariate analysis: a generalization of Wartenberg's multivariate spatial correlation. Geographical Analysis.
Smouse, P. & Peakall, R. (1999) Spatial autocorrelation analysis of individual multiallele and multilocus genetic structure. Heredity, 82, 561-573.

exemples

```
maf.xy <- mafragh$xy
maf.flo <- mafragh$flo
maf.listw <- nb2listw(neig2nb(mafragh$neig))
maf.coa <- dudi.coa(maf.flo, scannf = FALSE)
maf.randtest <- multispati.randtest(maf.coa, maf.listw)
maf.randtest
Monte-Carlo test
Observation: 0.2076
Call: multispati.randtest(dudi = maf.coa, listw = maf.listw)
Based on 999 replicates
Simulated p-value: 0.001
plot(maf.randtest)

clem.neig <- neig(n.line=15)
clem.nb <- neig2nb(clem.neig)
clem.listw <- nb2listw(clem.nb)
par(mfrow = c(1, 2))
clem.pca <- dudi.pca(clementines, scale = FALSE)
Select the number of axes: 2
clem.pca.ms <- multispati(clem.pca, clem.listw)
Select the first number of axes (>=1): 2
Select the second number of axes (>=0): 2
gearymoran(listw2mat(clem.listw), clem.pca$li)
```

```
class: krandtest
test number:   2
permutation number:   999
  test  obs    P(X<=obs) P(X>=obs)
1 Axis1 0.226  0.896     0.106
2 Axis2 -0.466 0.032     0.97

gearymoran(listw2mat(clem.listw), clem.pca.ms$li)
class: krandtest
test number:   4
permutation number:   999
  test obs    P(X<=obs) P(X>=obs)
1 CS1  0.423  0.972     0.03
2 CS2  0.668  1         0.001
3 CS19 -0.639 0.01      0.992
4 CS20 -0.508 0.034     0.968

clem.testglobal <- multispati.randtest(clem.pca, clem.listw)
plot(clem.testglobal)
```



R code

```
"multispati.randtest" <- function (dudi, listw, nrepet = 999) {
    if(!inherits(dudi,"dudi")) stop ("object of class 'dudi' expected")
    if(!inherits(listw,"listw")) stop ("object of class 'listw' expected")
    if(listw$style!="W") stop ("object of class 'listw' with style 'W' expected")

    "testmultispati"<- function(nrepet, nr, nc, tab, mat, lw, cw) {
        .C("testmultispati",
            as.integer(nrepet),
            as.integer(nr),
            as.integer(nc),
            as.double(as.matrix(tab)),
            as.double(mat),
            as.double(lw),
            as.double(cw),
            inersim=double(nrepet+1),
            PACKAGE="ade4")$inersim
    }

    tab<- dudi$tab
    nr<-nrow(tab)
    nc<-ncol(tab)
    mat<-listw2mat(listw)
    lw<- dudi$lw
    cw<- dudi$cw
    inersim<- testmultispati(nrepet, nr, nc, tab, mat, lw, cw)
    inertot<- sum(dudi$eig)
    inersim<- inersim/inertot
    obs <- inersim[1]
    w<-as.rtest(inersim[-1], obs, call = match.call())
    return(w)
}

####   ####   ####   ####   Fonction C
####   ####   ####   ####

void testmultispati (int *npermut, int *lig1, int *col1, double *tab, double *mat,
double *lw, double *cw, double *inersim)
{
/* Declarations de variables C locales */

    int         i, j, k, lig, col, nper, *numero;
    double      **X, **L, **Xperm;
    double      *d, *q, *dperm;

/* Allocation memoire pour les variables C locales */

    nper = *npermut;
    lig = *lig1;
```

```
    col= *col1;



    taballoc(&X, lig, col);
    taballoc(&L, lig, lig);
    taballoc(&Xperm, lig, col);
    vecintalloc (&numero, lig);
    vecalloc(&dperm, lig);
    vecalloc(&d, lig);
    vecalloc(&q, col);

/* On recopie les objets R dans les variables C locales */

    k = 0;
    for (j=1; j<=col; j++) {
        for (i=1; i<=lig; i++) {
            X[i][j] = tab[k];
            k = k + 1;
        }
    }

    k = 0;
    for (i=1; i<=lig; i++) {
        for (j=1; j<=lig; j++) {
            L[j][i] = mat[k];
            k = k + 1;
        }
    }

    k=0;
    for (i=1; i<=lig; i++) {
        d[i]=lw[k];
        k = k + 1;
    }

    k=0;
    for (i=1; i<=col; i++) {
        q[i]=cw[k];
        k = k + 1;
    }

/* On calcule la valeur observée */
    inersim[0]=traceXtdLXq(X, L, d, q);

/* On calcule les valeurs pour chaque simulation */

    for (j=1; j<=nper; j++) {
        getpermutation(numero, j);
        matpermut(X, numero ,Xperm);
        vecpermut(d, numero, dperm);
        inersim[j]=traceXtdLXq(Xperm, L, dperm, q);
    }


/* Libération des réservations locales */

freetab(X);
freetab(L);
freetab(Xperm);
freeintvec(numero);
freevec(dperm);
freevec(d);
freevec(q);
}

####   ####   ####   ####
####   ####   ####   ####
```

```
"multispati.rtest" <- function (dudi, listw, nrepet = 99) {
    if(!inherits(listw,"listw")) stop ("object of class 'listw' expected")
    if(listw$style!="W") stop ("object of class 'listw' with style 'W' expected")
    n = length(listw$weights)
    fun.lag = function (x) lag.listw(listw,x,TRUE)
    fun <- function (permuter = TRUE) {
        if (permuter) {
            permutation <- sample(n)
            y <- dudi$tab[permutation,]
            yw <- dudi$lw[permutation]
        } else {
            y <-dudi$tab
            yw <- dudi$lw
        }
        y <- as.matrix(y)
        ymoy = apply(y, 2, fun.lag)
        ymoy=ymoy*yw
        y = y*ymoy
        indexmoran <- sum(apply(y,2,sum)*dudi$cw)
        return(indexmoran)
    }
    inertot = sum(dudi$eig)
    obs <- fun (permuter = FALSE)/inertot
    if (nrepet == 0) return(obs)
    perm <- unlist(lapply(1:nrepet, fun))/inertot
    w <- as.rtest(obs = obs, sim = perm, call = match.call())
    return(w)
}
```

14. Construire un objet de la classe ‘phylog’

alias

newick2phylog, hclust2phylog, taxo2phylog, newick2phylog.addtools

description

Création d’objet de la classe ‘phylog’ pour manipuler des phylogénies. Les taxonomies et les classifications peuvent être vues comme des cas particuliers de structure phylogénétique.

- ‘newick2phylog’ assure la lecture d'un arbre phylogénétique au format 'Newick'. Le format newick peut être interprété de plusieurs manières. On se base sur :

http://evolution.genetics.washington.edu/phylip/newicktree.html
http://evolution.genetics.washington.edu/phylip/newick_doc.html

La fonction est systématiquement utilisée pour créer un objet de la classe 'phylog'.

- ‘hclust2phylog’ assure le passage d'un objet de la classe 'hclust' à une phylogénie.

- ‘taxo2phylog’ fait d'un objet 'taxo' un objet 'phylog'.

- ‘newick2phylog.addtools’ crée des objets pour intégrer les phylogénies dans les pratiques statistiques.

usage

```
newick2phylog (x.tre, add.tools = T, call = match.call())
hclust2phylog (hc, add.tools = TRUE)
taxo2phylog (taxo, add.tools = TRUE)
newick2phylog.addtools (res, tol = 1e-07)
```

arguments

- *x.tre* : est une chaîne de caractères codant un arbre au format 'Newick'
- *add.tools* : est une variable binaire. Si elle prend la valeur TRUE, les paramètres pour l'utilisation statistique de la phylogénie sont calculés. Appel interne de la fonction newick2phylog.addtools.
- *call* : ordre d'appel, utiliser la valeur par défaut
- *hc* : est un objet de la classe ‘hclust’
- *taxo* : est un objet de la classe ‘taxo’
- *res* : est un objet de la classe ‘phylog’
- *tol* : est une valeur seuil utilisée pour la définition des valeurs propres nulles

détails

La fonction accepte, à titre d'exercice, les structures pures du type :
"((((,,),,(,)),),(,));"
Les feuilles peuvent être labellisées partiellement ou totalement :
"((((aa,ab,ac),b,(ca,cb)),d),(e1,e2));"
Les nœuds peuvent être labellisés partiellement ou totalement:
"((((aa,ab,ac)N1,b,(ca,cb)N2)N3,d)N4,(e1,e2)N5)Root;"
L'arbre peut comporter des longueurs de branches :
"((((aa:0.1,ab:0.2,ac:0.1):5,b:5,(ca:0.1,cb:0.2):0.5):1,d:0.1):1,(e1:0.1,e2:0.2):0.6);"
Dans ce cas toutes les branches sauf la dernière doivent comporter une longueur.
Les étiquettes sont unquoted. Unquoted labels may not contain parentheses, square brackets, single_quotes, colons, semicolons, or commas.
Les blancs sont éliminés d'entrée ainsi que les commentaires entre crochets.
La chaîne peut être décomposée en éléments qui sont concaténés d'entrée.
Utiliser scan(file_name, character()) pour placer un fichier au format 'Newick' dans un vecteur.

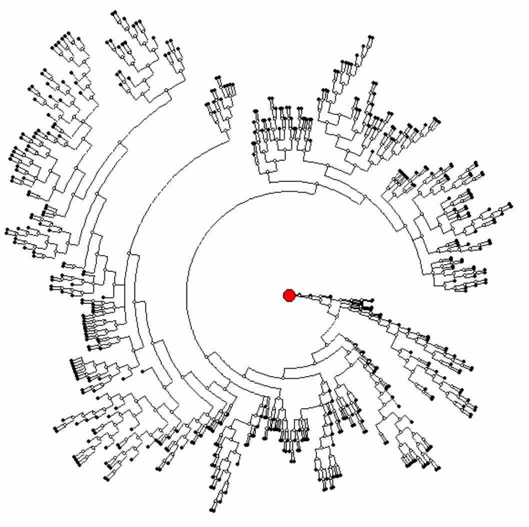
valeurs

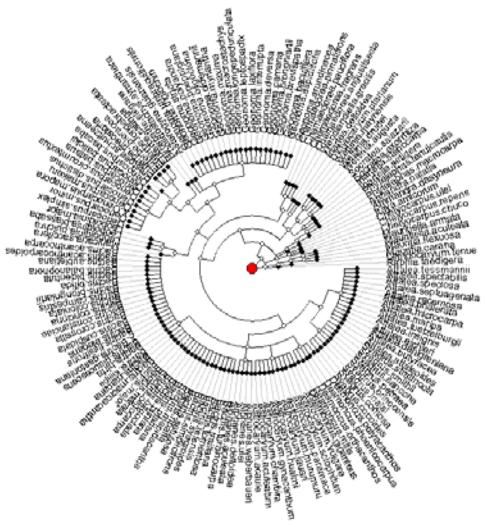
Une liste de la classe 'phylog'. Voir (Annexe). Si l'original n'est pas valué, toutes les longueurs de branches sont égales à 1.

exemples

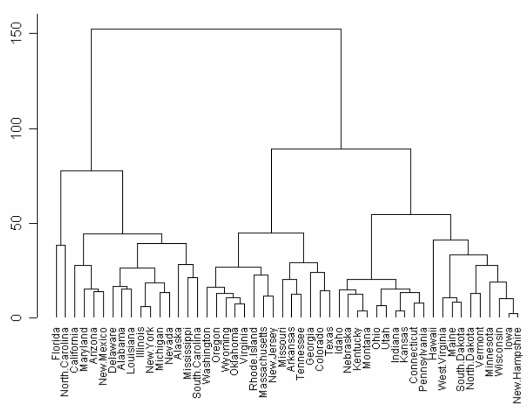
```
# newick2phylog
```

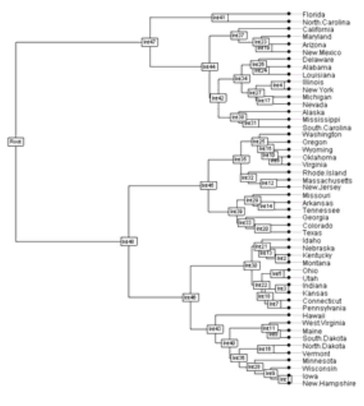
```
newick.try <-
c("(,((((((((((((((,,(,(,))),),(((,(,)),(,)),),(,(,)),(,)),(((((,(,)),),),(,)),((
((,((,),((,(,)),))),(,)),(,(,),,((,),(,)),)),((((((,),),(,(,))),),(,)),(((,),),))))
,((,,((,),)),(,)),((,),(,))),(((((((((,,),),,),),((,),)),(,),((,),)),),(((((,),),),
((,),)),(((,(,(,(,)))),(,)),(((,),((((((,),),),,),(,)),(,)),)),((,)",
",)))))),(,((,),(,)),((,(,)),)))),((((,(,(,))),((,(,)),,((,(,)),),)),(((,),),(((,),)
,))),((,),))),((((((((((,,,,(,)),),((,),)),(,(,))),(((((((((,(,)),(,)),((((,((,),(,
(,(,)))))),((,),(,(,)))),),((,),))),(((((((((,(,)),((,),(,))),),),),(((,((,),)),),((
,((,),)),)),(,)),(,(,(,)))),((((,(,)),(,)),(((,),(,)),(,),,(,))),(,))),(,,,))),((((
,),)",
",),(((,(,(,))),((,),)),(,)))),(,)),),(,((,(,)),),(((,),),)))),),(((,),),(,),(,(,)))
,(((,),(,)),((,),(,)))),(((,),((,),)),(((((,,,,),(,)),(,)),(,((,),))),))),(,(((((,
((((,(,)),),),)),),((,((,),((,((,),(,))),))),)),((((,),(((,),(,(,))),)),),)),((,),)
))),(((,((,,((,),)),)),),((,),))),((,),(,))),((,),)),(((((,),((,(,)),(((,(,)),(,(((
,),)",
",)))),))),(,),,),),),,(,)),((((,),,),),((,,,),((,),((,),))))),((((((,(,)),,(,)),,(,
),(,),),(((((,(,(,),)),(((,),,),(,))),),),),,,((,),)),),)),(((((,),(,(,)),),((,((,)
,),,),)),((((((,),((((,,,),(,(,))),(((,(,)),),(,))),)),),),),(,)),),),((,),))),((,
),)),(((((((((((,),),(((((,),),((,),)),(,)),),)),(,)),),(((((,),),(((,),),)),(,))
,),(,)),),),),),(,)),),((,),(,),,,)),(,(,(,)))),),(,)),),);")
phy1 <- newick2phylog(newick.try, FALSE)
# radial.phylog
radial.phylog(phy1, clabel.l = 0, circle = 2.2, clea = 0.5, cnod = 0.5)
radial.phylog(newick2phylog(newick.eg[[8]], FALSE),cnode = 1, clabel.l = 0.8)
```



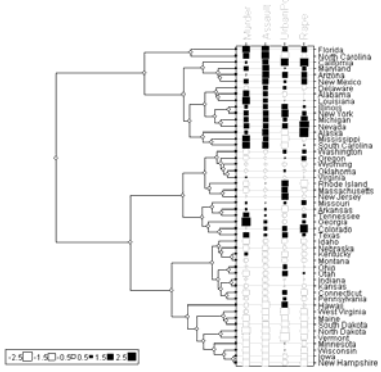


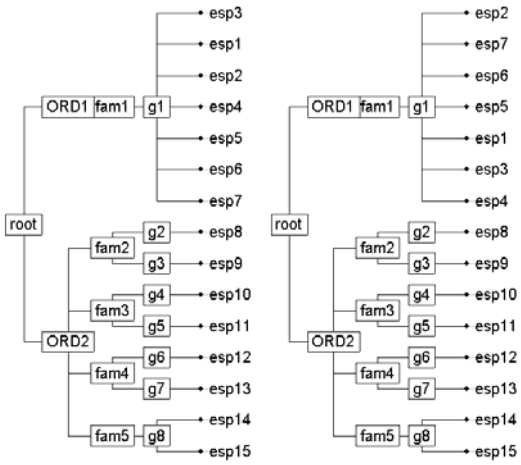
```
# hclust.phylog
data(USArrests)
help(USArrests)
hc <- hclust(dist(USArrests), "ave")
plot(hc, hang = -1)
phy <- hclust2phylog(hc)
plot.phylog(phy, clabel.l = 0.75, clabel.n = 0.6, f = 0.75)
```

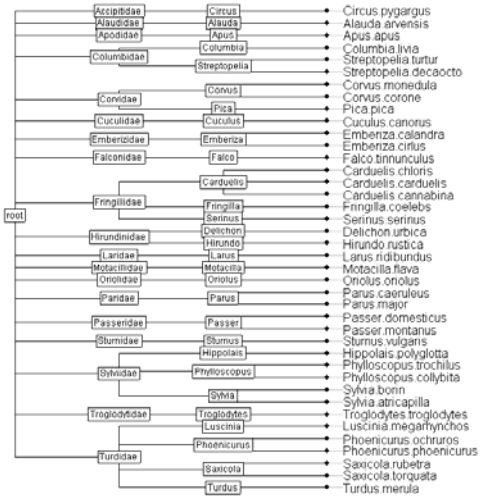




```
row.names(USArrests)
names(phy$leaves) #WARNING not the same for two reasons

row.names(USArrests) <- gsub(" ","_",row.names(USArrests))
row.names(USArrests)
names(phy$leaves) #WARNING not the same for one reason

USArrests <- USArrests[names(phy$leaves),]
row.names(USArrests)
names(phy$leaves) #the same

table.phylog(data.frame(scalewt(USArrests)),
 phy, csi = 2.5, clabel.r = 0.75, f = 0.7)

# taxo2phylog
tax <- as.taxo(taxo.eg[[1]])
tax.phy <- taxo2phylog(as.taxo(taxo.eg[[1]]))
par(mfrow = c(1,2))
plot.phylog(tax.phy, clabel.l = 1.25,
 clabel.n = 1.25, f = 0.75)
plot.phylog(taxo2phylog(as.taxo(
taxo.eg[[1]][sample(15),])),
 clabel.l = 1.25, clabel.n = 1.25, f = 0.75)

par(mfrow=c(1,1))
 plot.phylog(taxo2phylog(as.taxo(taxo.eg[[2]])),
  clabel.l = 1, clabel.n = 0.75, f = 0.65)
```







R code

```
"newick2phylog" <- function (x.tre, add.tools = TRUE, call =match.call()) {
    complete <- function(x.tre) {
        # Si la chaîne est en plusieurs morceaux elle est rassemblée
        if (length(x.tre) > 1) {
            w <- ""
            for (i in 1:length(x.tre)) w <- paste(w, x.tre[i],
                sep = "")
            x.tre <- w
        }
        # Si les parenthèses gauches et droites ont des effectifs différents -> out
        ndroite <- nchar(gsub("[^)]","",x.tre))
        ngauche <- nchar(gsub("[^(]","",x.tre))
        if (ndroite !=ngauche) stop (paste (ngauche,"( versus",ndroite,")"))
        # on doit trouver un ;
        if (regexpr(";", x.tre) == -1)
            stop("';' not found")
        # Tous les commentaires entre [] sont supprimés
        i <- 0
        kint <- 0
```

```
kext <- 0
arret <- FALSE
if (regexpr("\\[", x.tre) != -1) {
    x.tre <- gsub("\\[[^\\[]*\\]", "", x.tre, ext = FALSE)
}
x.tre <- gsub(" ", "", x.tre, ext = FALSE)
# On ne peut supprimer les . qui sont dans les distances !
# x.tre <- gsub("[.]","_", x.tre, ext = FALSE)
while (!arret) {
    i <- i + 1
    # examen de la chaîne par couple de charactères
    if (substr(x.tre, i, i) == ";")
        arret <- TRUE
    # (, c'est une feuille sans label
    if (substr(x.tre, i, i + 1) == "(,") {
        kext <- kext + 1
        add <- paste("Ext", kext, sep = "")
        x.tre <- paste(substring(x.tre, 1, i), add, substring(x.tre,
          i + 1), sep = "")
        i <- i + 1
    }
    # ,, c'est une feuille sans label
    else if (substr(x.tre, i, i + 1) == ",,") {
        kext <- kext + 1
        add <- paste("Ext", kext, sep = "")
        x.tre <- paste(substring(x.tre, 1, i), add, substring(x.tre,
          i + 1), sep = "")
        i <- i + 1
    }
    # ,) c'est une feuille sans label
    else if (substr(x.tre, i, i + 1) == ",)") {
        kext <- kext + 1
        add <- paste("Ext", kext, sep = "")
        x.tre <- paste(substring(x.tre, 1, i), add, substring(x.tre,
          i + 1), sep = "")
        i <- i + 1
    }
    # (: c'est une feuille sans label avec distance
    else if (substr(x.tre, i, i + 1) == "(:") {
        kext <- kext + 1
        add <- paste("Ext", kext, sep = "")
        x.tre <- paste(substring(x.tre, 1, i), add, substring(x.tre,
          i + 1), sep = "")
        i <- i + 1
    }
    # ,: c'est une feuille sans label avec distance
    else if (substr(x.tre, i, i + 1) == ",:") {
        kext <- kext + 1
        add <- paste("Ext", kext, sep = "")
        x.tre <- paste(substring(x.tre, 1, i), add, substring(x.tre,
          i + 1), sep = "")
        i <- i + 1
    }
    # ), c'est un noeud sans label
    else if (substr(x.tre, i, i + 1) == "),") {
        kint <- kint + 1
        add <- paste("I", kint, sep = "")
        x.tre <- paste(substring(x.tre, 1, i), add, substring(x.tre,
          i + 1), sep = "")
        i <- i + 1
    }
    # )) c'est un noeud sans label
    else if (substr(x.tre, i, i + 1) == "))") {
        kint <- kint + 1
        add <- paste("I", kint, sep = "")
        x.tre <- paste(substring(x.tre, 1, i), add, substring(x.tre,
          i + 1), sep = "")
        i <- i + 1
```

```
    }
    # ): c'est un noeud sans label avec distance
    else if (substr(x.tre, i, i + 1) == "):") {
        kint <- kint + 1
        add <- paste("I", kint, sep = "")
        x.tre <- paste(substring(x.tre, 1, i), add, substring(x.tre,
          i + 1), sep = "")
        i <- i + 1
    }
    # ); c'est la racine sans label
    else if (substr(x.tre, i, i + 1) == ");") {
        add <- "Root"
        x.tre <- paste(substring(x.tre, 1, i), add, substring(x.tre,
          i + 1), sep = "")
        i <- i + 1
    }
}
# extraction de l'information non structurelle
lab.points <- strsplit(x.tre, "[(),;]")[[1]]
lab.points <- lab.points[lab.points != ""]
# recherche de la présence des longueurs
no.long <- (regexpr(":", lab.points) == -1)
# si il n'y avait aucune longueur
if (all(no.long)) {
    lab.points <- paste(lab.points, ":", c(rep("1", length(no.long) -
        1), "0.0"), sep = "")
}
# si il y en vait partout sauf à la racine
else if (no.long[length(no.long)]) {
    lab.points[length(lab.points)] <- paste(lab.points[length(lab.points)],
        ":0.0", sep = "")
}
# si il y en a et il n'y en a pas -> out
else if (any(no.long)) {
    print(x.tre)
    stop("Non convenient data leaves or nodes with and without length")
}
w <- strsplit(x.tre, "[(),;]")[[1]]
w <- w[w != ""]
leurre <- make.names(w, unique = TRUE)
leurre <- gsub("[.]","_", leurre, ext = FALSE)
for (i in 1:length(w)) {
    old <- paste(w[i])
    x.tre <- sub(old, leurre[i], x.tre, ext = FALSE)
}
# extraction des labels et des longueurs
w <- strsplit(lab.points, ":")
label <- function(x) {
    # ici on peut travailler sur les labels
    lab <- x[1]
    lab <- gsub("[.]","_", lab, ext = FALSE)
    return (lab)
}

longueur <- function(x) {
    long <- x[2]
    return (long)
}

labels <- unlist(lapply(w, label))
longueurs <- unlist(lapply(w, longueur))
# ici on peut travailler sur les labels
labels <- make.names(labels, TRUE)
labels <- gsub("[.]","_", labels, ext = FALSE)
w <- labels
for (i in 1:length(w)) {
    new <- w[i]
    x.tre <- sub(leurre[i], new, x.tre, ext = FALSE)
```

```
        }
        # on les a remis à leur place
        cat <- rep("", length(w))
        for (i in 1:length(w)) {
            new <- w[i]
            if (regexpr(paste(")", new, sep = ""), x.tre, ext = FALSE) !=
                -1)
                cat[i] <- "int"
            else if (regexpr(paste(",", new, sep = ""), x.tre,
                ext = FALSE) != -1)
                cat[i] <- "ext"
            else if (regexpr(paste("(", new, sep = ""), x.tre,
                ext = FALSE) != -1)
                cat[i] <- "ext"
            else cat[i] <- "unknown"
        }
        return(list(tre = x.tre, noms = labels, poi = as.numeric(longueurs),
            cat = cat))
    }
    res <- complete(x.tre)
    poi <- res$poi
    nam <- res$noms
    names(poi) <- nam
    cat <- res$cat
    res <- list(tre = res$tre)
    res$leaves <- poi[cat == "ext"]
    names(res$leaves) <- nam[cat == "ext"]
    res$nodes <- poi[cat == "int"]
    names(res$nodes) <- nam[cat == "int"]
    nleaves <- length(res$leaves)
    nnodes <- length(res$nodes)
    listclass <- list()
    dnext <- c(names(res$leaves), names(res$nodes))
    listpath <- as.list(dnext)
    names(listpath) <- dnext
    x.tre <- res$tre
    while (regexpr("[(]", x.tre) != -1) {
        a <- regexpr("([^()]*)", x.tre, ext = FALSE)
        n1 <- a[1] + 1
        n2 <- n1 - 3 + attr(a, "match.length")
        chasans <- substring(x.tre, n1, n2)
        chaavec <- paste("(", chasans, ")", sep = "")
        nam <- unlist(strsplit(chasans, ","))
        w1 <- strsplit(x.tre, chaavec, ext = FALSE)[[1]][2]
        parent <- unlist(strsplit(w1, "[,);]", ext = FALSE))[1]
        listclass[[parent]] <- nam
        x.tre <- gsub(chaavec, "", x.tre, ext = FALSE)
        w2 <- which(unlist(lapply(listpath, function(x) any(x[1] ==
            nam))))
        for (i in w2) {
            listpath[[i]] <- c(parent, listpath[[i]])
        }
    }
    res$parts <- listclass
    res$paths <- listpath
    dnext <- c(res$leaves, res$nodes)
    names(dnext) <- c(names(res$leaves), names(res$nodes))
    res$droot <- unlist(lapply(res$paths, function(x) sum(dnext[x])))
    res$call <- call
    class(res) <- "phylog"
    if (!add.tools) return(res)
    return(newick2phylog.addtools(res))
}

####   ####   ####   ####
####   ####   ####   ####

"hclust2phylog" <- function (hc, add.tools = TRUE) {
```

```
    if (!inherits(hc, "hclust"))
        stop("'hclust' object expected")
    labels.leaves <- make.names(hc$labels, TRUE)
    nleaves <- length(labels.leaves)
    nnodes <- nrow(hc$merge)
    labels.nodes <- paste("Int", 1:nnodes, sep = "")
    l.bra <- matrix("$", nnodes, 2)
    for (i in nnodes:1) {
        for (j in 1:2) {
            if (hc$merge[i, j] < 0)
                l.bra[i, j] <- as.character(hc$height[i])
            else l.bra[i, j] <- as.character(hc$height[i] - hc$height[hc$merge[i,
                j]])
        }
    }
    l.eti <- matrix("$", nnodes, 2)
    for (i in nnodes:1) {
        for (j in 1:2) {
            if (hc$merge[i, j] > 0)
                l.eti[i, j] <- labels.nodes[hc$merge[i, j]]
            else l.eti[i, j] <- labels.leaves[-hc$merge[i, j]]
        }
    }
    tre <- paste("(", l.eti[nnodes, 1], ":", l.bra[nnodes, 1],
        ",", l.eti[nnodes, 2], ":", l.bra[nnodes, 2], ")Root:0.0;",
        sep = "")
    for (j in (nnodes - 1):1) {
        w <- paste("(", l.eti[j, 1], ":", l.bra[j, 1], ",", l.eti[j,
            2], ":", l.bra[j, 2], ")", labels.nodes[j], ":",
            sep = "")
        tre <- gsub(paste(labels.nodes[j], ":", sep = ""), w,
            tre, ext = FALSE)
    }
    res <- newick2phylog(tre, add.tools, call=match.call())
    return(res)
}

####   ####   ####   ####
####   ####   ####   ####

"taxo2phylog" <- function (taxo, add.tools = TRUE) {
    if (!inherits(taxo, "taxo"))
        stop("Object 'taxo' expected")
    nr <- nrow(taxo)
    nc <- ncol(taxo)
    leaves.names <- row.names(taxo)
    res <- paste("root;")
    x <- taxo[, nc]
    xred <- as.character(levels(x))
    w <- "("
    for (i in xred) w <- paste(w, i, ",", sep = "")
    res <- paste(w, ")", res, sep = "")
    res <- sub(",)", ")", res, ext = FALSE)

    for (j in nc:2) {
        x <- taxo[, j]
        y <- taxo[, j - 1]
        for (k in 1:nlevels(x)) {
            w <- "("
            old <- as.character(levels(x)[k])
            yred <- unique(y[x == levels(x)[k]])
            yred <- levels(y)[yred]
            for (i in yred) w <- paste(w, i, ",", sep = "")
            w <- paste(w, ")", old, sep = "")
            w <- sub(",)", ")", w, ext = FALSE)
            res <- sub(old, w, res, ext = FALSE)
        }
    }
```

```
    x <- taxo[, 1]
    y <- leaves.names
    for (k in 1:nlevels(x)) {
        w <- "("
        old <- as.character(levels(x)[k])
        yred <- y[x == levels(x)[k]]
        for (i in yred) w <- paste(w, i, ",", sep = "")
        w <- paste(w, ")", old, sep = "")
        w <- sub(",)", ")", w, ext = FALSE)
        res <- sub(old, w, res, ext = FALSE)
    }
    return(newick2phylog(res, add.tools, call=match.call()))
}

####   ####   ####   ####
####   ####   ####   ####

"newick2phylog.addtools" <- function(res, tol =1e-07) {
    require(ade4)

    nleaves <- length(res$leaves) # nombre de feuilles
    nnodes <- length(res$nodes)    # nombre de noeuds
    node.names <- names(res$nodes) # noms des feuilles
    leave.names <- names(res$leaves) # noms des noeuds
    dimnodes<-unlist(lapply(res$parts,length)) # nombres de descendants immédiats
de chaque noeud
    effnodes <- dimnodes # recevra le nombre de descendants total de chaque noeud
    wnodes <- lgamma(dimnodes+1)
    # recevra le logarithme du nombre de permuations compatibles
    # avec la sous-arborescence associée à chaque noeud



    # les matrices de proximité #
    a <- matrix(0, nleaves, nleaves)
    ia <- as.numeric(col(a))
    ja <- as.numeric(row(a))
    a <- cbind(ia, ja)[ia < ja, ]
    # a contient la liste des couples de feuilles
    floc1 <- function(x) {
        # x est un couple de numéros de deux feuilles i, avec i<j
        # Cette fonction renvoie
        # resw - la distance à la racine du premier ancêtre commun de deux feuilles
        # resa - l'inverse des produits des nombres de descendants des noeuds
        # rencontrés sur le plus court chemin entre les deux feuilles
        c1 <- rev(res$paths[[x[1]]])
        c2 <- rev(res$paths[[x[2]]])
        commonnodes <- c1[c1 %in% c2]
        resw <- res$droot[commonnodes[1]]
        d1 <- c1[! (c1 %in% c2)][-1]
        d2 <- c2[! (c2 %in% c1)][-1]
        pathij <- c(d1,d2,commonnodes[1])
        resa <- 1/prod(unlist(dimnodes[pathij]))
        return(c(resw,resa))
    }
    b <- apply(a, 1, floc1)
    names(b) <- NULL
    w <- matrix(0, nleaves, nleaves)
    w[col(w) < row(w)] <- b[1,]
    w <- w + t(w)
    diag(w) <- res$droot[leave.names]
    dimnames(w) <- list(leave.names,1:nleaves)

    res$Wmat <- w

    ############################
    # la composante Wmat contient la matrice W des distances racine-premier ancêtre
commun
```

```
##############################

w <- diag(res$Wmat)
w <- matrix(w, nleaves, nleaves)
w <- w + t(w) - 2 * res$Wmat
w <- mat2dist(sqrt(w))
attr(w, "Labels") <- leave.names

res$Wdist <- w
##############################
# la composante Wdist contient la matrice des racines des distances nodales
# qui forment une distance euclidienne
##############################
w <- res$Wmat
w <- w / sum(w)
w <- bicenter.wt(w)
w <- eigen(w,sym=TRUE)
res$Wvalues <- w$values[-nleaves]*nleaves
w <- as.data.frame(w$vectors[,-nleaves]*sqrt(nleaves))
row.names(w) <- leave.names
names(w) = paste("W",1:(nleaves-1),sep="")
res$Wscores <- w


w <- matrix(0, nleaves, nleaves)
w[col(w) < row(w)] <- b[2,]
w <- w + t(w)
# On rajoute la diagonale pour que A soit bistochastique
floc1 <- function(x) {
    # cette fonction renvoie pour une feuille la fréquence des représentations
    # compatibles qui placent cette feuille tout en haut ou tout en bas
    c1 <- rev(res$paths[[x]])
    c1 <- c1[-1] # premier ancetre, second ancetre, ..., racine
    resw <- dimnodes[c1] # ordre des noeuds
    resw <- 1/prod(unlist(resw))
    return(resw)
}
diag(w) <- unlist(lapply(leave.names,floc1))
dimnames(w) <- list(leave.names,1:nleaves)
res$Amat <- w
##############################
# la composante Amat contient la matrice des probabilités
# pour une feuille d'être juste au dessus d'une autre
# dans l'ensemble des permutations compatibles
##############################
# double centrage
w <- bicenter.wt(w)
# diagonalisation
eig <- eigen (w, sym = TRUE)
w0 <- abs(eig$values)/max(abs(eig$values))
w0 <- which(w0<tol)
if (length(w0)==0) stop ("abnormal output : no null eigenvalue")
if (length(w0)==1) w0 <- (1:nleaves)[-w0]
else if (length(w0)>1) {
    # on ajoute le vecteur dérivé de 1n
    w <- cbind(rep(1,nleaves),eig$vectors[,w0])
    # on orthonormalise l'ensemble
    w <- qr.Q(qr(w))
    # on met les valeurs propres à 0
    eig$values[w0] <- 0
    # on remplace les vecteurs du noyau par une base orthonormée contenant
    # en première position le parasite
    eig$vectors[,w0] <- w[,-ncol(w)]
    # on enlève la position du parasite
    w0 <- (1:nleaves)[-w0[1]]
}
rank <- length(w0)
res$Avalues <- eig$values[w0]*nleaves
```

```
    ##############################
    # la composante Avalues contient les valeurs propres de QAQ
    ##############################
    res$Adim <- sum(res$Avalues>tol)
    ##############################
    # la composante Adim contient le nombre de valeurs propres positives
    # associées à la composante positive de la variance
    ##############################
    w <- eig$vectors[,w0]*sqrt(nleaves)
    w <- data.frame(w)
    row.names(w) <- leave.names
    names(w) <- paste("A",1:rank,sep="")
    res$Ascores <- w
    ##############################
    # la composante Ascores contient une base orthobormée de l'orthogonal de n
    # pour la pondération uniforme. Elle définit un phylogramme
    ##############################

    # Complément : la valeur des noeuds #

    floc1 <- function(k) {
        # k est un numéro de noeud
        # x est un vecteur comportant un nom de noeud et des noms de descendants
        # de ce noeud.
        # A la fin parts wnodes contient le logarithme
        # du nombre de permutations compatibles de chaque sous-arbre
        # et effnodes contient le nombre de descendants de chaque sous-arbre
        y <- res$parts[[k]]
        x <- y[y%in%names(res$nodes)]
        n1 <- names(res$parts)[k]
        if (length(x)<=0) return(NULL)
        effnodes[n1] <<- effnodes[n1] - length(x) + sum(effnodes[x])
        wnodes[n1] <<- wnodes[n1] + sum(wnodes[x])
        return(NULL)
    }

    lapply(1:length(res$parts),floc1)
    # typolo.value <- 1-exp(wnodes-lgamma(effnodes+1)) abandon

    ####res$Aparam  <-  data.frame(x1=I(dimnodes),  x2=I(effnodes),  x3=I(wnodes),
x4=I(typolo.value))
    res$Aparam <- data.frame(ndir=dimnodes, nlea=effnodes, lnperm=wnodes)
    ##############################
    # la composante Aparam est un data.frame de paramètre sur l'ensemble des noeuds
    # x1 = nombre de descendants directs
    # x2 = nombre de feuilles descendantes
    # x3 = log du nombre de permutations compatibles avec la phylogénie extraite
    #  x4  =  1-rapport  du  nombre  de  permutations  compatibles  sur  le  nombre  de
permutations totales
    # pour la phylogénie extraite dans ce noeud
    # cet indice vaut 0 si le noeud est final et est maximal à la racine
    # attention il ne vaut pas 1 mais 1-epsilon quand il est affiché 1
    ##############################

    # Complément : la base B #
    w1 <- matrix(0, nleaves, nnodes)
    ####x1 <- res$Aparam$x2 #le nombre de feuilles descendantes
    x1 <- res$Aparam$lnperm #on trie sur le log des permutations
    #  on  calcule  une  matrice  auxiliaire  pour  avoir  la  liste  des  feuilles
descendantes
    # pour chacun des noeuds
    dimnames(w1) <- list(leave.names, names(x1))
    for (i in leave.names) {
        ancetres <- res$paths[[i]]
        ancetres <- rev(ancetres)[-1]#rev(ancetres[-1])[-1]
        w1[i, ancetres] <- 1
    }
    w1 <- cbind(w1, diag(1, nleaves))
```

```
    dimnames(w1)[[2]] <- c(names(x1),leave.names)
    x1 <- c(x1, rep(-1,nleaves))
    names(x1) <-dimnames(w1)[[2]]
    # La matrice w1 contient 1 en i-j si la feuille i descend du noeud j

    ######################################
    # on construit une famille d'indicatrices de classes
    # Une arête de l'arborescence est un lien de descendance
    # Chaque noeud et chaque feuille (à l'expection de la racine) a un seul
ascendant
    # Il y a n+f-1 arêtes. Le noeud j a m(j) descendants
    # Les feuilles n'en n'ont pas. Donc m(1)+m(2)+ ... + m(n) = n+f-1
    # Il y a n+f-1 arêtes réparties en n blocs.
    # Il y a donc n+f-1-n=f-1 descendants indicateurs DI quand on enlève une arête
descendante par noeud
    # Rien n'est conservé pour un noeud avec un seul descendant
    # Pour chaque DI on utilise l'indicatrice de la classe des feuilles descendant
de cet noeud
    # la composante Bindica contient f-1 indicatrices de classes de feuilles
    # names (w) contient des noms de descendants
    # nomuni contient les noms de DI pour l'étiquetage final
    ######################################
    funnoe <- function (noeud) {
        # renvoie pour un noeud une liste dont chaque composante est un descendant
immédiat du noeud
        # caractérisé par la liste des feuilles qui en descendent sous forme de
matrice
        # d'indicatrices. Le dernier descendant immédiat du noeud est éliminé.
        x <- res$parts[[noeud]] # les descendants immédiats
        xval <- x1[x] # le nombre de feuilles descendantes des descendants
        xval <- rev(sort(xval)) # triée
        x <- names(xval) # on récupère lesquels
        x <- x[-length(x)] # on enlève le dernier
        if (length(x) ==0) return(NULL)
        if              (length(x)              ==1)              xmat              <-
matrix(w1[,x],ncol=1,dimnames=list(leave.names,noeud))
        else {
           xmat <- w1[,x]
           dimnames(xmat)[[2]] <- rep(noeud, ncol(xmat))
        }
        return (list(xmat, x))
        # les noms des colonnes de xmat repète le nom du noeud
        # dans y on a le nom des descendants retenus
    }

    nomuni <- NULL
    w <- matrix(1,nleaves,1)
    dimnames(w) <- list(leave.names, "un")
    for (i in names(x1)[1:nnodes]) {
        provi <- funnoe(i)
        if (!is.null(provi)) {
            w <-cbind(w, provi[[1]])
            nomuni <- c(nomuni,provi[[2]])
        }
    }
    w <- w[,-1]
    nomrepet <- dimnames(w)[[2]]
    names(nomrepet) <- nomuni
    dimnames(w)[[2]] <- nomuni
    names(nomuni) <- nomuni
    ############################
    # Les indicatrices sont classées par ordre décroissant
    # de xtQWQx la variance phylogénétique formelle de l'indicatrice centrée
    # Bindica n'a qu'une valeur pédagogique et ne sert pas explicitement
    # mais la procédure est simple
    # 1) définition des indicatrices, il y en a toujours f-1
    # 2) rangement par valeur décroissante de la forme quadratique
    # Ce rangement est conservé dans res$Bindica
```

```
    # les valeurs du critère de rangement dans Bvalues
    # 3) rajout de 1n devant
    # 4) orthonormalisation
    # on obtient toujours une base orthonormée de l'orthogonal de 1n
    ############################
    floc <- function (x) {
        x <- x-mean(x)
        sum(t(res$Wmat*x)*x)
    }
    # chaque indicatrice donne une valeur
    ### w.val <- unlist(apply(w,2, floc))
    w.val <- x1[nomuni]
    # trie par ordre descendant
    w.val <- rev(sort(w.val))
    # lesquels
    w <- w[,names(w.val)]
    # nomrepet / w sont triés
    nomrepet <- nomrepet[names(w.val)]
    res$Bindica <- as.data.frame(w)
    w <- cbind(rep(1,nleaves),w)
    w <- qr.Q(qr(w))
    w <- w[, -1] * sqrt(nleaves)
    w <- data.frame(w)
    row.names(w) <- leave.names
    names(w) <- paste("B",1:(nleaves-1),sep="")
    res$Bscores <- w
    ### res$Bvalues <- w.val
    lw <- lapply(node.names, function (x) which(nomrepet==x))
    names(lw) <- node.names
    fun1 <- function (x) {
        if (length(x)==0) return("x")
        if (length(x)==1) return(as.character(x))
        y <- x[1]
        for(k in 2:length(x)) y <- paste(y,x[k],sep="/")
        return(y)
    }
    lw <- unlist(lapply(lw, fun1))
    res$Blabels <- lw
    return(res)
}
```

15. Bases orthonormées

alias

orthobasis.mat
orthobasis.listw
orthobasis.neig
orthobasis.wavelet
orthobasis.line
orthobasis.circ
print.orthobasis

description

Ces fonctions définissent des bases de vecteurs propres **D**-orthonormés. Pour l'instant, seule la pondération uniforme $\mathbf{D} = diag(1/n, ..., 1/n)$ a été envisagée. Les fonctions 'orthobasis.neig', 'orthobasis.mat', 'orthobasis.listw', 'orthobasis.line', 'orthobasis.circ' définissent une base orthonormée associée au spectre de la matrice d'un graphe. La fonction 'orthobasis.wavelet' introduit les bases orthonormées associées aux analyses en ondelettes. La fonction 'print.orthobasis' édite les principales caractéristiques de chaque base orthonormée.

usage

```
orthobasis.mat(mat, cnw = TRUE)
orthobasis.neig(neig)
orthobasis.wavelet(n, wf.name)
orthobasis.line(n)
orthobasis.circ(n)
print.orthobasis(orthobas)
```

arguments

- *mat* : est la matrice binaire **M** associée à un graphe de voisinage
- *cnw* : si TRUE, le graphe est complété afin que chaque unité statistique ait le même nombre de voisins (constant neighboring weight)
- *neig* : est un objet de la classe 'neig'
- *n* : est un entier correspondant au nombre d'unités statistiques
- *wf.name* : est une chaîne de caractères correspondant au nom de la famille d'ondelettes condidérées ('haar' : ondelettes d'Haar, 'd4' : ondelettes de Daubechies(4), …)
- *orthobas* : est un objet de la classe 'orthobasis'

détails

Un couple arbitraire d'une matrice symétrique **A** et d'une pondération **D** définissent une (ou plusieurs) base(s) **D**-orthonormale(s) de vecteurs **A**-orthogonaux en colonnes dans une matrice **U** *n-n* qui vérifie :

$$\begin{cases} \mathbf{U}^t\mathbf{D}\mathbf{U} = \mathbf{I}_n \\ \mathbf{U}^t\mathbf{A}\mathbf{U} = diag(\lambda_1, ..., \lambda_n) \end{cases}$$

En statistique, ce qui se passe sur une variable constante, d'un point de vue typologique, est généralement sans intérêt (elle a une variance nulle, une autocovariance nulle, une variance locale nulle, une corrélation nulle avec n'importe quoi, elle n'a pas de composante phylogénétique, …). Au lieu de préciser qu'on ne fera un calcul de forme bilinéaire que sur des variables centrées, autant prendre directement la forme qui a les mêmes valeurs pour toute variable centrée et qui prend la valeur 0 pour les variables constantes. On ajoutera donc à la décomposition canonique de la matrice **A** la notion de **D**-centrage. On dira que la matrice **A** est **D**-centrée si tous les vecteurs propres sont **D**-orthogonaux au vecteur $\mathbf{1}_n$. En particulier, si $\mathbf{1}_n$ est un vecteur propre de **A** alors la matrice **A** est **D**-centrée. Sinon, on récupère les vecteurs propres **D**-orthogonaux au vecteur $\mathbf{1}_n$ par orthonormalisation de Gram-Schmidt. Dans tous les cas, on élimine le vecteur $\mathbf{1}_n$ de la base orthonormée et l'on travaille sur des variables centrées pour la pondération **D**. La base obtenue n'a donc que *n-1* vecteurs.

La fonction 'orthobasis.mat' définit une base orthonormée à partir d'une matrice symétrique **A** quelconque. **A** peut être la matrice binaire d'un graphe de voisinage ou bien **A** peut la matrice d'une forme bilinéaire symétrique. La base est définie par

la diagonalisation de la matrice $\left(\mathbf{Id}_n - \mathbf{1}_n\mathbf{1}_n^t\mathbf{D}\right)^t\left(\mathbf{A}/\mathbf{1}_n^t\mathbf{A}\mathbf{1}_n\right)\left(\mathbf{Id}_n - \mathbf{1}_n\mathbf{1}_n^t\mathbf{D}\right)/n$. Lorsque l'option cnw = TRUE, la pondération de voisinage est uniforme donc les valeurs propres sont aussi bien des indices de Moran que des indices de Geary.

La fonction 'orthobasis.neig' définit une base orthonormée associée à l'opérateur de lissage définit par Méot. On diagonalise l'opérateur de voisinage $(\mathbf{P}-\mathbf{F})\mathbf{D}$ avec $\mathbf{F} = \mathbf{A}/\mathbf{1}_n^t\mathbf{A}\mathbf{1}_n$ et $\mathbf{P} = diag(\mathbf{F}\mathbf{1}_n)$ puis l'on calcule à partir de là les valeurs propres de l'opérateur de lissage et l'on ordonne les vecteurs propres par variance locale croissante.

Les fonctions 'orthobasis.line', 'orthobasis.circ' définissent également une base orthonormée associée à l'opérateur de lissage introduit par Méot à partir des solutions analytiques.

valeurs

Ces fonctions retournent des objets de la classe 'orthobasis', sous classe de la classe 'data.frame'. Un objet de la classe 'orthobasis' est donc un data frame avec en colonnes les n-1 vecteurs propres orthonormées. Ses attributs sont :

- *$names* : est un vecteur dont chaque élément définit le nom d'un vecteur propre
- *$row.names* : est un vecteur dont chaque élément définit le nom d'une unité statistique
- *$class* : est un vecteur à deux éléments correspondant à la sous classe et la classe de l'objet
- *$values* : est un vecteur correspondant aux valeurs propres de la matrice diagonalisée
- *$weights* : est un vecteur correspondant à la pondération **D** envisagée
- *$call* : rappelle la commande qui a permis de créer cet objet

références

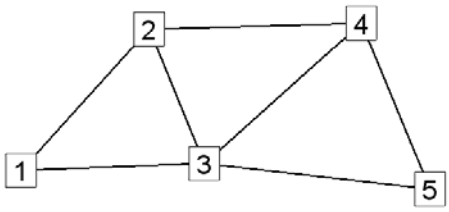
Cornillon, P.-A. (1998) Prise en compte de proximités en analyse factorielle et comparative. Thèse, Ecole Nationale Supérieure Agronomique, Montpellier.
Cvetkoviv, D.M., Doob, M., & Sachs, H. (1979) Spectra of graphs Academic Press, New York.
Méot, A., Chessel, D., & Sabatier, D. (1993). Opérateurs de voisnage et analyse des données spatio-temporelles. In Biométrie et environnement (eds J.D. Lebreton & B. Asselain). Masson.
Percival, D.B. & Walden, A.T. (2000) Wavelet Methods for Time Series Analysis Cambridge University Press.

voir également

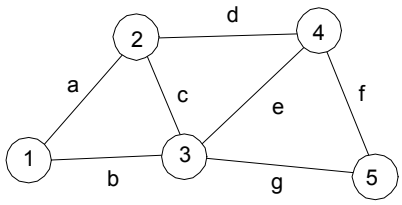
gridrowcol
orthogram
orthobasis2kfbs

exemples

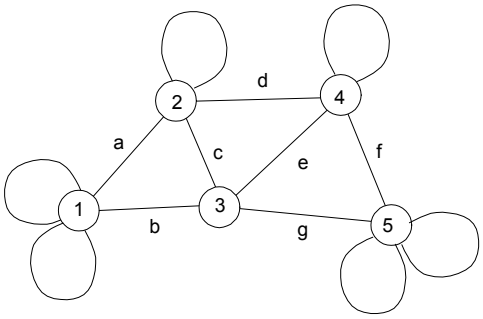
```
# matrice de voisinage binaire
w <- matrix(c(1,1,2,2,3,3,4,2,3,3,4,4,5,5), 7,2)
sim.neig <- neig(edges = w)
x <- c(37,97,123,197,229)
y <- c(49,115,52,118,40)
sim.xy <- data.frame(x,y)
s.label(sim.xy, neig = sim.neig, inc = FALSE,
 grid = FALSE, addaxes = FALSE, clab = 2)
```



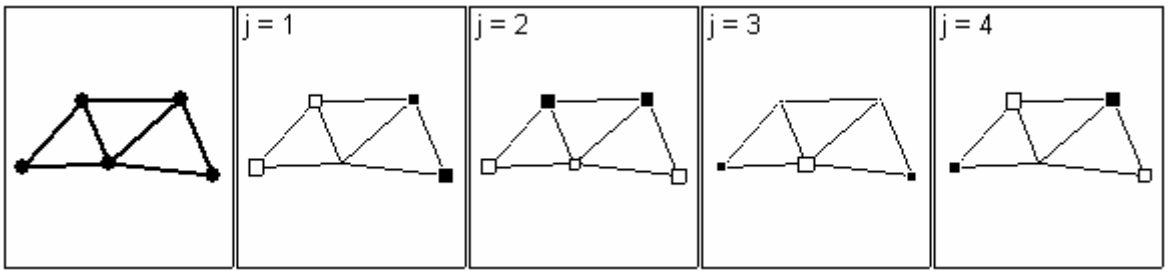

```
mat <- neig2mat(sim.neig)
print(mat)
  1 2 3 4 5
1 0 1 1 0 0
2 1 0 1 1 0
3 1 1 0 1 1
4 0 1 1 0 1
5 0 0 1 1 0

"cnw" <- function(mat) {
             margi <- apply(mat,1,sum)
```



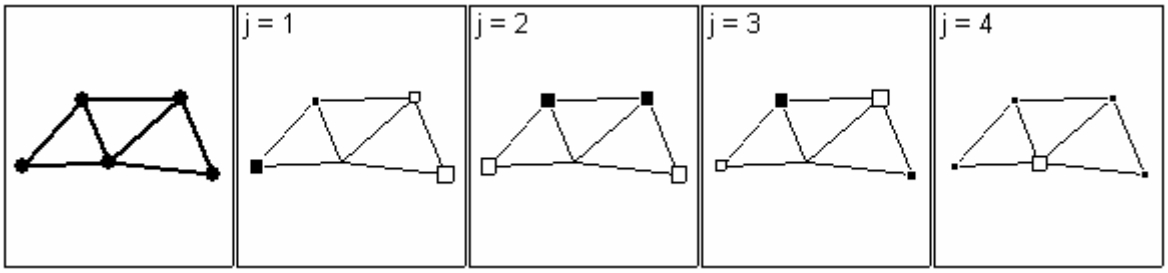
```
            margi <- max(margi)-margi
            mat <- mat + diag(margi)
            print(mat)
    }

 cnw(mat)
  1 2 3 4 5
1 2 1 1 0 0
2 1 1 1 1 0
3 1 1 0 1 1
4 0 1 1 1 1
5 0 0 1 1 2
```




```
      # vecteurs propres
orthobas <- orthobasis.mat(mat, cnw = FALSE)
par(mfrow = c(1, 5))
par(mar = c(0.1,0.1,0.1,0.1))
s.label(sim.xy, neig = sim.neig, inc = FALSE,
 grid = FALSE, addaxes = FALSE, clab = 0, cpoint = 4)
for(i in 1:4){
s.value(sim.xy, orthobas[,i], neig = sim.neig, grid = FALSE, addaxes = FALSE,
 inc = FALSE, cleg = 0, sub = paste("j =", i, sep = " "), csub = 2)
}
```

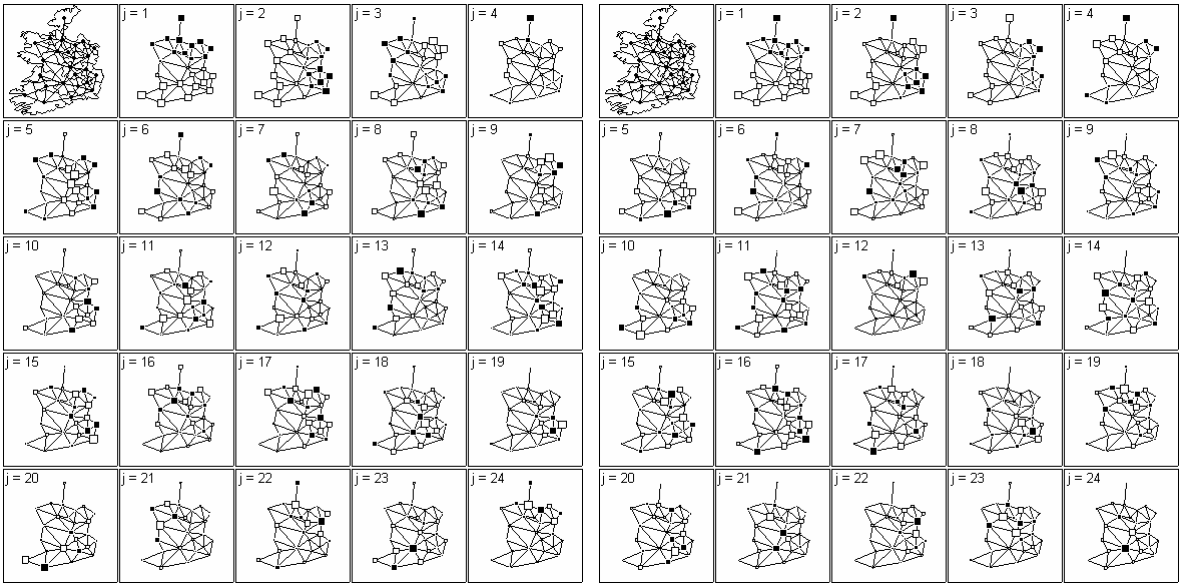



```
orthobas <- orthobasis.mat(mat, cnw = TRUE)
par(mfrow = c(1, 5))
par(mar = c(0.1,0.1,0.1,0.1))
s.label(sim.xy, neig = sim.neig, inc = FALSE,
 grid = FALSE, addaxes = FALSE, clab = 0, cpoint = 4)
for(i in 1:4){
s.value(sim.xy, orthobas[,i], neig = sim.neig, grid = FALSE, addaxes = FALSE,
 inc = FALSE, cleg = 0, sub = paste("j =", i, sep = " "), csub = 2)
}
```



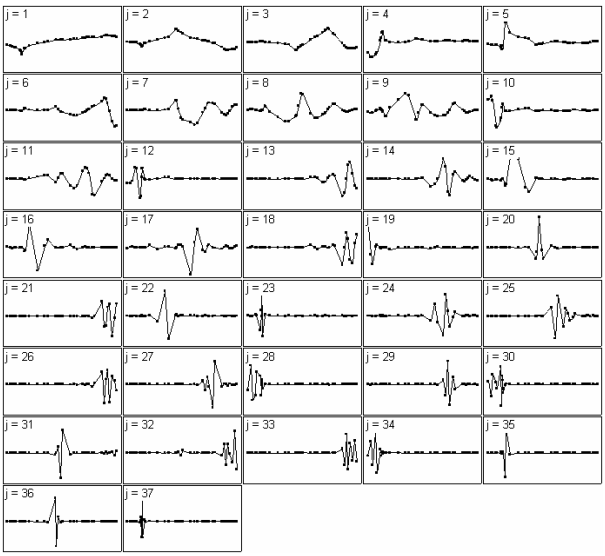

```
# matrice de pondération de voisinage
ir.xy <- irishdata$xy.utm
ir.area <- irishdata$area.utm
ir.neig <- neig(area = irishdata$area.utm)
ir.nb <- neig2nb(ir.neig)
ir.list.w <- apply(irishdata$link.utm, 1, function(x) x[x!=0])
listw <- nb2listw(ir.nb, glist = ir.list.w, style = "W")
par(mar = c(0.1, 0.1, 0.1, 0.1))
plot(c(0, 0), type = "n", ylab = "", asp = 1,
 xaxt = "n",  yaxt = "n", frame.plot = FALSE,
 xlim = c(30,290), ylim = c(5770,6110))
s.label(ir.xy, neig = ir.neig, cneig = 2, area = ir.area,
 addaxes = F, grid = F, add.plot = T, clab = 1)
```

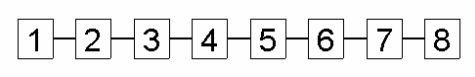
```
      # vecteurs propres de l'opérateur de voisinage
orthobas <- orthobasis.mat(listw2mat(listw))
par(mfrow = c(5, 5))
par(mar = c(0.1, 0.1, 0.1, 0.1))

plot(c(0, 0), type = "n", ylab = "", asp = 1,
 xaxt = "n",  yaxt = "n", frame.plot = FALSE,
 xlim = c(30,290), ylim = c(5770,6110))
s.label(ir.xy, neig = ir.neig, cneig = 1, area = ir.area,
 addaxes = F, grid = F, add.plot = T, clab = 0, cpoint = 2)

for(i in 1:24){
plot(c(0, 0), type = "n", ylab = "", asp = 1,
 xaxt = "n",  yaxt = "n", frame.plot = FALSE,
 xlim = c(30,288), ylim = c(5770,6110))
s.value(ir.xy, orthobas[,i], neig = ir.neig, cneig = 0.5,
addaxes = F, grid = F, add.plot = T, cleg = 0,
 sub = paste("j =", i, sep = " "), csub = 2)
}

orthobas <- orthobasis.neig(ir.neig)
par(mfrow = c(5, 5))
par(mar = c(0.1, 0.1, 0.1, 0.1))

plot(c(0, 0), type = "n", ylab = "", asp = 1,
 xaxt = "n",  yaxt = "n", frame.plot = FALSE,
 xlim = c(30,290), ylim = c(5770,6110))
s.label(ir.xy, neig = ir.neig, cneig = 1, area = ir.area,
 addaxes = F, grid = F, add.plot = T, clab = 0, cpoint = 2)

for(i in 1:24){
plot(c(0, 0), type = "n", ylab = "", asp = 1,
 xaxt = "n",  yaxt = "n", frame.plot = FALSE,
 xlim = c(30,288), ylim = c(5770,6110))
s.value(ir.xy, orthobas[,i], neig = ir.neig, cneig = 0.5,
addaxes = F, grid = F, add.plot = T, cleg = 0,
 sub = paste("j =", i, sep = " "), csub = 2)
}
```

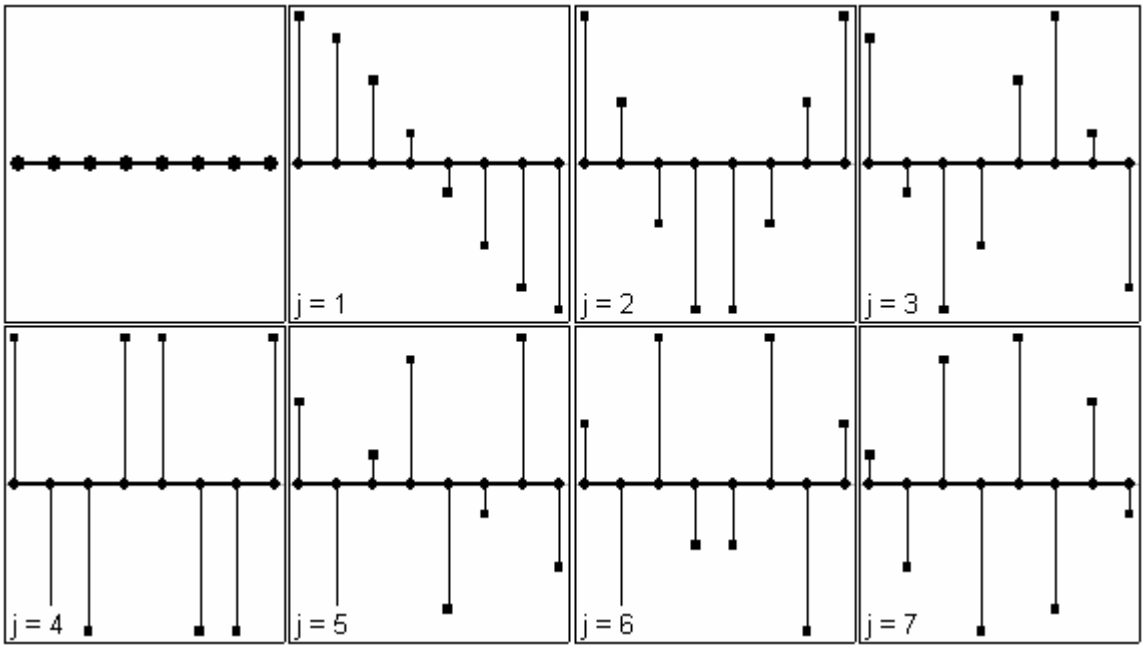



```
fac <- guadeloupe$coord
fac <- as.factor(fac)
table(fac)
y <- guadeloupe$zooplankton
y.mean <- lapply(split(y, fac), mean)
eti <- names(y.mean)
```

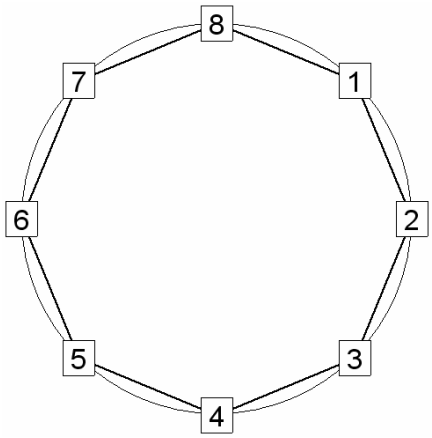
```
coord <- as.numeric(eti)
biomass <- unlist(y.mean)
xy <- cbind.data.frame(coord, rep(0, 38))
names(xy) <- c("x", "y")
mat <- dist2mat(dist(xy))
mat <- 1/mat
diag(mat) <- rep(0, 38)
mat <- mat/apply(mat, 1, sum)
mat <- as.data.frame(mat)
names(mat) <- eti
row.names(mat) <- eti
mat <- as.matrix(mat)
listw <- mat2listw(mat)
listw$style <- "W"
orthobas <- orthobasis.listw(listw)

par(mfrow = c(8, 5))
par(mar = c(0.1,0.1,0.1,0.1))

for(i in 1:37){
      dotchart.line(orthobas[,i], coord, axe1 = FALSE, axe2 = FALSE, csub = 0, cdot
= 0.65, joining = F, ranging = TRUE, yranging = c(-5, 6))
      scatterutil.sub(paste("j =", i, sep = " "), csub = 2, "topleft")
      box()

}

# graphe linéaire
n <- 8
ligne <- neig(n.line = n)
x <- 1 :8
y <- rep(0,8)
xy <- cbind.data.frame(x, y)
names(xy) <- c("x", "y")
par(mar = c(0.1, 0.1, 0.1, 0.1))
plot(c(0, 0), type = "n", ylab = "", asp = 1,
 xaxt = "n",  yaxt = "n", frame.plot = FALSE,
 xlim = c(0,9), ylim = c(-1,1))
s.label(xy, neig = ligne, cneig = 2,
 addaxes = F, grid = F, add.plot = T, clab = 2)

      # vecteurs propres de l'opérateur de voisinage
orthobas <- orthobasis.line(n)
par(mfrow = c(2, 4))
par(mar = c(0.1,0.1,0.1,0.1))

plot(c(0, 0), type = "n", ylab = "", asp = 1,
 xaxt = "n",  yaxt = "n", frame.plot = FALSE,
 xlim = c(0.9,8.1), ylim = c(-1,1))
s.label(xy, neig = ligne, cneig = 2,
 addaxes = F, grid = F, add.plot = T, clab = 0, cpoint = 4)

for(i in 1:7){
      dotchart.line(orthobas[,i], axe1 = FALSE, axe2 = FALSE, csub = 0, cdot = 1)
      #scatterutil.sub(paste("j =", i, sep = " "), csub = 2, "topleft")
      s.label(xy, neig = ligne, cneig = 2, addaxes = F, grid = F, add.plot = T,
        clab = 0, cpoint = 3, sub = paste("j =", i, sep = " "), csub = 2)
      box()
}
```



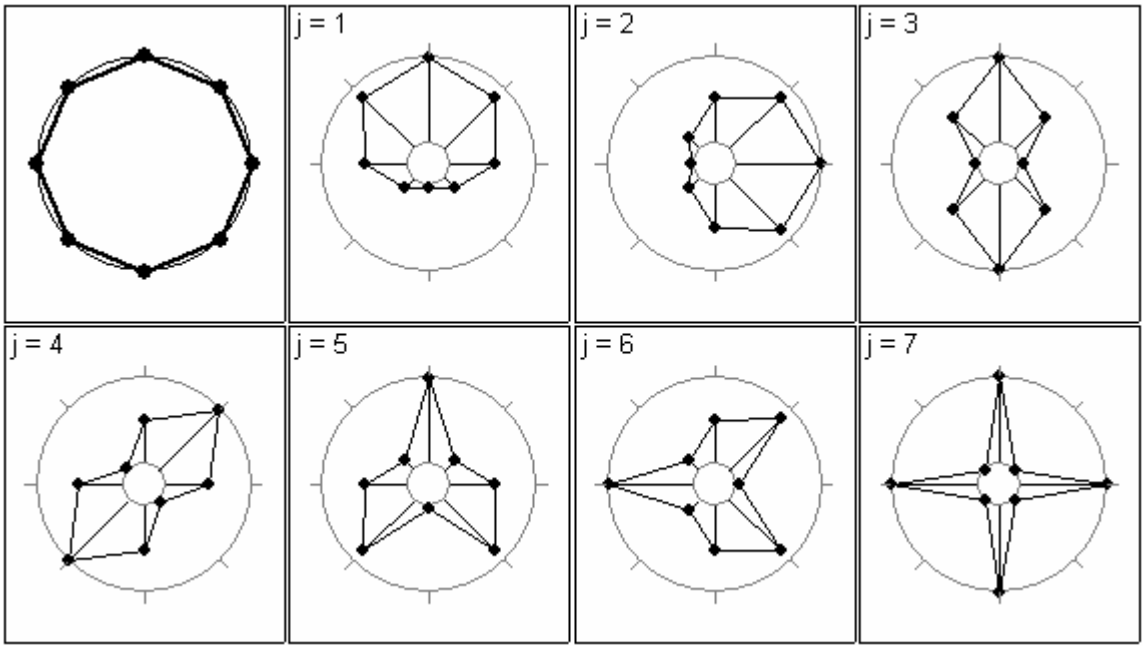





```
# graphe circulaire
n <- 8
cercle <- neig(n.circle = n)
alpha <- pi/2 - (1:n) * 2 * pi/n
z <- rep(1, 8)
x <- z * cos(alpha)
y <- z * sin(alpha)
xy <- cbind.data.frame(x, y)
names(xy) <- c("x", "y")
par(mar = c(0.1, 0.1, 0.1, 0.1))
plot(c(0, 0), type = "n", ylab = "", asp = 1,
 xaxt = "n",  yaxt = "n", frame.plot = FALSE,
 xlim = c(-1.2, 1.2), ylim = c(-1.2, 1.2))
symbols(0, 0, cir = 1, inc = FALSE, add = TRUE)
s.label(xy, neig = cercle, cneig = 2,
 addaxes = F, grid = F, add.plot = T, clab = 2)
```




```
      # vecteurs propres de l'opérateur de voisinage
orthobas <- orthobasis.circ(n)
par(mfrow = c(2, 4))
par(mar = c(0.1,0.1,0.1,0.1))

plot(c(0, 0), type = "n", ylab = "", asp = 1,
 xaxt = "n",  yaxt = "n", frame.plot = FALSE,
 xlim = c(-1.2, 1.2), ylim = c(-1.2, 1.2))
symbols(0, 0, cir = 1, inc = FALSE, add = TRUE)
s.label(xy, neig = cercle, cneig = 2, cpoint = 4,
 addaxes = F, grid = F, add.plot = T, clab = 0)

for(i in 1:7){
      dotcircle(orthobas[,i], cleg = 0)
      scatterutil.sub(paste("j =", i, sep = " "), csub = 2, "topleft")
      box()
}
```



R code

```
"orthobasis.mat" <- function(mat, cnw = TRUE) {
    if (!is.matrix(mat)) stop ("matrix expected")
    if (any(mat<0)) print ("negative value in 'mat'")
    if (nrow(mat) != ncol(mat)) stop ("squared matrix expected")
    mat <- (mat+t(mat))/2
    nlig <- nrow(mat)
    if (is.null(dimnames(mat))) {
        w <- paste("P",1:nrow(mat),sep="")
        dimnames(mat) <- list(w,w)
    }
    labels <- dimnames(mat)[[1]]
    if (cnw) {
        margi <- apply(mat,1,sum)
        margi <- max(margi)-margi
        mat <- mat + diag(margi)
    }
    mat <- mat/sum(mat)
    wt <- rep ((1/nlig),nlig)
    # calculs extensibles à une pondération quelconque
    wt <- wt/sum(wt)
    # si mat wt est la pondération marginale associée à mat
    # tot = sum(mat)
    # mat = mat-matrix(wt,nlig,nlig,byrow=TRUE)*wt*tot
    # encore plus particulier mat = mat-1/nlig/nlig
    # en général les précédents sont des cas particuliers
    U <- matrix(1,nlig,nlig)
    U  <- diag(1,nlig)-U*wt
    mat <- U%*%mat%*%t(U)
    wt <- sqrt(wt)
    mat <- t(t(mat)/wt)
    mat <- mat/wt
    eig <- eigen(mat,sym=TRUE)
    w0 <- abs(eig$values)/max(abs(eig$values))
    tol <- 1e-07
    w0 <- which(w0<tol)
    if (length(w0)==0) stop ("abnormal output : no null eigenvalue")
    else if (length(w0)==1) w0 <- (1:nlig)[-w0]
    else if (length(w0)>1) {
        # on ajoute le vecteur dérivé de 1n
        w <- cbind(wt,eig$vectors[,w0])
        # on orthonormalise l'ensemble
        w <- qr.Q(qr(w))
        # on met les valeurs propres à 0
```

```
        eig$values[w0] <- 0
        # on remplace les vecteurs du noyau par une base orthonormée contenant
        # en première position le parasite
        eig$vectors[,w0] <- w[,-ncol(w)]
        # on enlève la position du parasite
        w0 <- (1:nlig)[-w0[1]]
    }
    mat <- eig$vectors[,w0]/wt
    mat <- data.frame(mat)
    row.names(mat) <- labels
    names(mat) <- paste("S",1:(nlig-1),sep="")
    attr(mat,"values") <- eig$values[w0]
    attr(mat,"weights") <- rep(1/nlig,nlig)
    attr(mat,"call") <- match.call()
    attr(mat,"class") <- c("orthobasis","data.frame")
    return(mat)
}

####    ####    ####    ####
####    ####    ####    ####

"orthobasis.listw" <- function(listw) {
    appel = match.call()
    if(!inherits(listw,"listw")) stop ("object of class 'listw' expected")
    if(listw$style!="W") stop ("object of class 'listw' with style 'W' expected")
    n = length(listw$weights)
    fun <- function (x) {
        num = listw$neighbours[[x]]
        wei = listw$weights[[x]]
        res = rep(0,n)
        res[num] = wei
        return (res)
    }
    b0 <- matrix(unlist(lapply(1:n,fun)),n,n)
    b0=(t(b0)+b0)/2
    b0=bicenter.wt(b0)
    a0 <- eigen(b0, sym = TRUE)
    #barplot(a0$values)
    a0 <- a0$vectors
    a0 <- cbind(rep(1,n),a0)
    a0 <- qr.Q(qr(a0))
    a0 <- as.data.frame(a0[,-1])*sqrt(n)
    row.names(a0) <- attr(listw,"region.id")
    names(a0) <- paste("VP", 1:(n-1), sep = "")
    z <- apply(a0,2,function(x) sum((t(b0*x)*x))/n)
    attr(a0,"values") <- z
    attr(a0,"weights") <- rep(1/n,n)
    attr(a0,"call") <- appel
    attr(a0,"class") <- c("orthobasis","data.frame")
    return(a0)
}

####    ####    ####    ####
####    ####    ####    ####

"orthobasis.neig" <- function(neig) {
    appel = match.call()
    if(!inherits(neig,"neig")) stop ("object of class 'neig' expected")
    n <- length(attr(neig,"degree"))
    m <- sum(attr(neig,"degree"))
    poivoisi <- attr(neig,"degree")/m
    if (is.null(names(poivoisi))) names(poivoisi) <- as.character(1:n)
    d0 = neig2mat(neig)
    d0 = diag(poivoisi)-d0/m
    eig <- eigen(d0, sym = TRUE)
    tol <- 1e-07
    w0 <- abs(eig$values)/max(abs(eig$values))
    w0 <- which(w0<tol)
```

```
    if (length(w0)==0) stop ("abnormal output : no null eigenvalue")
    else if (length(w0)==1) w0 <- (1:n)[-w0]
    else if (length(w0)>1) {
        # on ajoute le vecteur dérivé de 1n
        wt <- rep(1,n)
        w <- cbind(wt,eig$vectors[,w0])
        # on orthonormalise l'ensemble
        w <- qr.Q(qr(w))
        # on met les valeurs propres à 0
        eig$values[w0] <- 0
        # on remplace les vecteurs du noyau par une base orthonormée contenant
        # en première position le parasite
        eig$vectors[,w0] <- w[,-ncol(w)]
        # on enlève la position du parasite
        w0 <- (1:n)[-w0[1]]
    }
    w0 <- rev(w0)
    valpro <- eig$values[w0]
    eig <- eig$vectors[,w0]
    eig <- as.data.frame(eig)*sqrt(n)
    z <- apply(eig,2,function(x) sum(x*x*poivoisi))
    z <- z - valpro*n
    w <- rev(order(z))
    z <- z[w]
    eig <- eig[,w]
    row.names(eig) <- names(poivoisi)
    names(eig) <- paste("VP", 1:(n-1), sep = "")
    attr(eig,"values") <- z
    attr(eig,"weights") <- rep(1/n,n)
    attr(eig,"call") <- appel
    attr(eig,"class") <- c("orthobasis","data.frame")
    return(eig)
}

####    ####    ####    ####
####    ####    ####    ####

"orthobasis.wavelet" <- function(n, wf.name){

    if (!require(waveslim)) stop ("Please install waveslim")

# on vérifie que n est une puissance de 2
appel <- match.call()
J <- log(n)/log(2)  #nombre de niveau
b <- floor(J)
if ((J-b)^2 > 1e-10) stop ("n is not a power of 2")

# on définit de manière récursive la base en commençant par le filtre passe haut de
# plus bas niveau
filter.seq <- "H" #filtre correspondant au niveau 1
h <- wavelet.filter(wf.name = wf.name, filter.seq = filter.seq)
l.h <- length(h) # si la longueur du filtre est supérieur à n, il faut utiliser un
# filtre circulaire périodique
lag.h <- 2  # définit la translation nécessaire à la définition de vecteurs
# orthogonaux à partir du filtre h
dim.h <- 2**(J-1)   # définit le nombre de vecteur du sous-espace associé au filtre
# h
names.h <- paste(rep("B1", dim.h), 1:dim.h, sep = ".") # définit le nom des
# vecteurs du sous-espace
res.h <- matrix(0, nrow = n, ncol = dim.h)

res <- NULL
names.res <- NULL

for(j in 1:J){
    if (l.h > n){
         l <- floor(l.h - 1)
         x <- rep(0, l*n)
```

```
         x[1:l.h] <- h
         x <- matrix(x, byrow = TRUE, ncol = n)
         x <- apply(x, 2, sum)
         y <- n
    }
        else {
            x <- h
            y <- l.h
        }
    k <- 1:y

    for(i in 1:dim.h){
        res.h[k,i] <- x
        k <- k + lag.h - floor((k + lag.h - 1)/n)*n
    }
    res <- cbind(res.h, res)
    names.res <- c(names.h, names.res)
    filter.seq <- paste(filter.seq, "L", sep = "")
    h <- wavelet.filter(wf.name = wf.name, filter.seq = filter.seq)
    l.h <- length(h)
    lag.h <- 2**(j + 1)
    dim.h <- 2**(J-j-1)
    names.h <- paste(rep(paste("B", j + 1, sep = ""), dim.h), 1:dim.h, sep = ".")
    res.h <- matrix(0, nrow = n, ncol = dim.h)
}

res <- res*sqrt(n)
res <- res[n:1,]
res <- data.frame(res)
row.names(res) <- paste("u", 1:n, sep = "")
names(res) <- names.res
attr(res,"values") <- NULL
attr(res,"weights") <- rep(1/n,n)
attr(res,"call") <- appel
attr(res,"class") <- c("orthobasis","data.frame")
return(res)
}

####    ####    ####    ####
####    ####    ####    ####

"orthobasis.line" <- function (n) {
    appel <- match.call()
    res <- NULL
    r2 <- sqrt(2)
    for (k in 1:(n-1)) {
        x <- cos(k*pi*(2*(1:n)-1)/2/n)
        x <- sqrt(n)*x/sqrt(sum(x*x))
        res <-c(res,x)
    }
    res <- matrix(res,n)
    res <- data.frame(res)
    row.names(res) <- paste("u",1:n,sep="")
    names(res) <- paste("B",1:(n-1),sep="")
    w <- (1:(n-1))*pi/2/n
    valpro <- 4*(sin(w)^2)/n
    poivoisi <- c(1,rep(2,n-2),1)
    poivoisi <- poivoisi/sum(poivoisi)
    norm <- unlist(apply(res, 2, function(a) sum(a*a*poivoisi)))
    y <- valpro*n*n/2/(n-1)
    val <- norm - y
    attr(res,"values") <- val
    attr(res,"weights") <- rep(1/n,n)
    attr(res,"call") <- appel
    attr(res,"class") <- c("orthobasis","data.frame")
    return(res)
}
```

```
####    ####    ####    ####
####    ####    ####    ####

"orthobasis.circ" <- function (n) {
    appel <- match.call()
    if (n < 3) stop ("'n' too small")
    "vecprosin" <- function(k) {
        x <- sin(2*k*pi*(1:n)/n)
        x <- x/sqrt(sum(x*x))
    }
    "vecprocos" <- function(k) {
        x <- cos(2*k*pi*(1:n)/n)
        x <- x/sqrt(sum(x*x))
    }
    "valpro" <- function(k, bis = TRUE) {
        x <- (4/n)*((sin(k*pi/n))^2)
        if (bis) x <- c(x,x)
        return(x)
    }

    k <- floor(n/2)
    if (k == n/2) {
        #n est pair
        w1 <- matrix(unlist(lapply(1:k,vecprocos)),n,k)
        w2 <- matrix(unlist(lapply(1:(k-1),vecprosin)),n,k-1)
        res <- cbind(w1,w2)
        res[,seq(1,2*k-1,by = 2)] <- w1
        res[,seq(2,2*k-2,by = 2)] <- w2
        vp <- unlist(lapply(1:(k-1),valpro))
        vp <- c(vp, valpro(k,FALSE))
    } else {
        # n est impair
        w1 <- matrix(unlist(lapply(1:k,vecprocos)),n,k)
        w2 <- matrix(unlist(lapply(1:k,vecprosin)),n,k)
        res <- cbind(w1,w2)
        res[,seq(1,2*k-1,by = 2)]<-w1
        res[,seq(2,2*k,by = 2)]<-w2
        vp <- unlist(lapply(1:k,valpro))
    }
    res <- sqrt(n)*res
    res <- as.data.frame(res)
    row.names(res) <- paste("u",1:n,sep="")
    names(res) <- paste("B",1:(n-1),sep="")
    attr(res,"values") <- 1 - n*vp/2
    attr(res,"weights") <- rep(1/n,n)
    attr(res,"call") <- appel
    attr(res,"class") <- c("orthobasis","data.frame")
    return(res)
}

####    ####    ####    ####
####    ####    ####    ####

"print.orthobasis" <- function(orthobas) {
    if (!inherits(orthobas,"orthobasis")) stop ("for 'orthobasis' object")
    cat("Orthonormal basis: ")
    n <- nrow(orthobas)
    p <- ncol(orthobas)
    if (n!=(p+1)) stop ("Non convenient dimension: author's error")
    cat("data.frame with",n,"rows and",ncol(orthobas),"columns\n")
    cat("-----------------------------------\n")
    cat("Columns are an orthonormal basis of 1n-orthogonal for\n")
    cat("the inner product defined by the weights attribute\n")
    cat("-----------------------------------\n")
    w <- attributes(orthobas)
    if (!is.null(w$"names")) cat("names =", w$names[1],"...",w$names[p],"\n")
    if            (!is.null(w$"row.names"))            cat("row.names           =",
w$row.names[1],"...",w$row.names[n],"\n")
```

```
    if              (!is.null(w$"weights"))              cat("weights              =",
w$weights[1],"...",w$weights[n],"\n")
    if (!is.null(w$"values")) cat("values =", w$values[1],"...",w$values[p],"\n")
    if (!is.null(w$"class")) cat("class =", w$class,"\n")
    if (!is.null(w$"call")) {
        cat("call =")
        print(w$"call")
    }
}
```

16. Décomposition de la variance par les vecteurs d'une base orthonormée

alias

orthogram

description

'orthogram' est une fonction permettant le calcul de la décomposition de la variance d'une variable par les vecteurs d'une base orthonormée. Elle donne une représentation graphique de la décomposition et de la décomposition cumulée. Elle met en œuvre quatre tests non paramétriques contre l'absence de structure. Les statistiques sont dérivées de la décomposition de la variance.

usage

```
orthogram(x, orthobas = NULL, neig = NULL, phylog = NULL, nrepet = 999, posinega =
0, na.action = c("fail", "mean"), cdot = 1.5, cfont.main = 1.5, lwd = 2, nclass,
high.scores = 0)
```

arguments

- *x* : est un vecteur correspondant à la variable quantitative dont on veut analyser la variance
- *orthobas* : est un objet de la classe 'orthobasis'
- *neig* : est un objet de la class 'neig'
- *phylog* : est un objet de la classe 'phylog'
- *nrepet* : est un entier représentant le nombre de permutations pour les tests non paramétriques
- *posinega* : est un entier positif ou nul. Ce paramètre intervient dans la définition du test non paramétrique basée sur le rapport de variance. S'il prend la valeur 0, le test n'est pas mis en œuvre
- *na*.action : si cet argument prend la valeur 'fail', la fonction plante lorsque la variable présente des valeurs manquantes. Sinon, on remplace les valeurs manquantes par la moyenne des valeurs
- *cdot* : est un entier positif représentant la taille des points sur le graphique de la variance cummulée
- *cfont.main* : est un entier positif représentant la taille des caractères des titres
- *lwd* : est un entier positif représentant la taille des lignes en pointillés
- *nclass* : est un entier positif représentant le nombre de classes utilisé pour la représentation des histogrammes
- *high.scores* : est un entier positif ou nul. S'il prend une valeur *k* supérieure à 0, la fonction retourne les *k* noms des vecteurs pour lesquels le carré de corrélation avec la variable est maximal.

détails

Chaque variable $\tilde{\mathbf{x}}$ admet une décomposition unique sur les vecteurs de la base **B** :

$$\tilde{\mathbf{x}} = \sum_{i=1}^{n-1} r_i \mathbf{b}_i \cdot$$

Les r_i correspondent aux corrélations entre la variable et chaque vecteur de la base **B**. La variance d'une variable se décompose alors en somme de carrés de corrélation :

$$\left\|\tilde{\mathbf{x}}\right\|_{\mathbf{D}} = 1 = \tilde{\mathbf{x}}^t \mathbf{D} \tilde{\mathbf{x}} = (\mathbf{Br})^t \mathbf{DBr} = \mathbf{rB}^t \mathbf{DBr} = \mathbf{r}^t \mathbf{r} = \sum_{i=1}^{n} r_i^2 \cdot$$

La fonction retourne une représentation des r_i et des $\sum_{j=1}^{i} r_j^2$ en fonction de *i*. Les enveloppes de confiance sont déterminées par permutation des valeurs de la variable.

Quatre tests non paramétriques contre l'absence de structure sont envisagés. Ils sont basés sur les statistiques :

$$\mathrm{R2Max}(\tilde{\mathbf{x}}) = \max\left(r_1^2, \ldots, r_{t-1}^2\right) \;;\; D\max(\tilde{\mathbf{x}}) = \max_{1 \le m \le t-1}\left(\sum_{i=1}^{m} r_i^2 - \frac{m}{t-1}\right) \;;\; \mathrm{SkR2k}(\tilde{\mathbf{x}}) = \sum_{i=1}^{t-1} i r_i^2 \;;\; \mathrm{SCE}(\tilde{\mathbf{x}}) = \sum_{i=1}^{t-1}\left(r_i^2 - r_{i-1}^2\right)^2$$

valeurs

'orthogram' retourne un objet de la classe 'krandtest' correspondant aux résultats des quatres tests non paramétriques.

Si la paramètre 'high.scores' prend une valeur *k* supérieure à 0, la fonction retourne les *k* noms des vecteurs pour lesquels le carré de corrélation avec la variable est maximal.

références

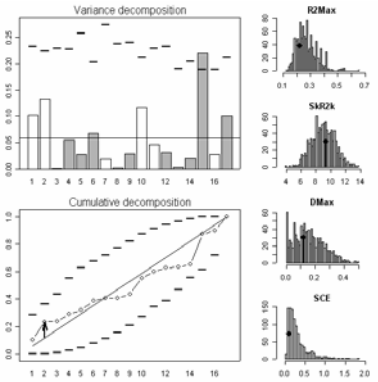
Ollier, S., Couteron, P., & Chessel, D. (soumis) Orthonormal transforms to describe and test the phylogenetic signal. Biometrics.
Percival, D.B. & Walden, A.T. (2000) Wavelet Methods for Time Series Analysis Cambridge University Press.

voir également

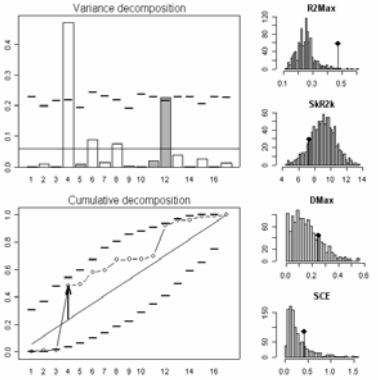
gridrowcol
orthobasis
mld
val.kfbs

exemples

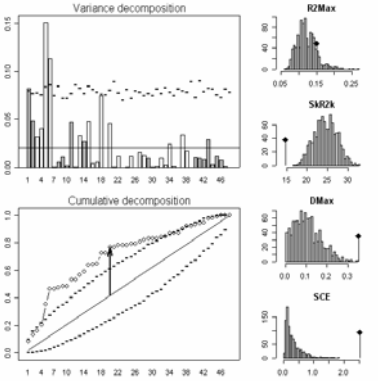
```
# un exemple avec une phylogénie : absence de structure
procella.phy <- newick2phylog(
 procella.phy$tre)
procella.trait<-procella$traits$ALE
procella.trait<-procella$traits$ALE[
 !is.na(procella.trait)]
procella.trait <- sqrt(procella.trait)
orthogram(procella.trait,
 phylog = procella.phy)
class: krandtest
test number:    4
permutation number:    999
  test   obs    P(X<=obs) P(X>=obs)
1 R2Max 0.22   0.257     0.751
2 SkR2k 9.341  0.6       0.402
3 Dmax  0.117  0.383     0.619
4 SCE   0.095  0.145     0.857
```



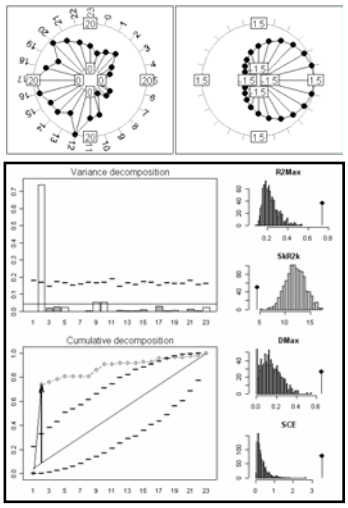

```
# un autre exemple avec une phylogénie : structure à un seul niveau
ung.phy <- newick2phylog(ungulates$tre)
afbw <- log(ungulates$tab$afbw)
orthogram(afbw, phylog = ung.phy)
class: krandtest
test number:    4
permutation number:    999
  test   obs    P(X<=obs) P(X>=obs)
1 R2Max 0.47   0.993     0.009
2 SkR2k 7.29   0.13      0.872
3 Dmax  0.247  0.815     0.187
4 SCE   0.415  0.821     0.181
```




```
# un dernier exemple : structure diffuse
mjrochet.phy <- newick2phylog(mjrochet$tre)
tab <- log((mjrochet$tab))
tab0 <- data.frame(scalewt(tab))
orthogram(tab0[,1], mjrochet.phy$Bscores)
class: krandtest
test number:    4
permutation number:    999
  test   obs    P(X<=obs) P(X>=obs)
1 R2Max 0.15   0.88      0.122
2 SkR2k 14.71  0.001     1
3 Dmax  0.352  1         0.001
4 SCE   2.527  1         0.001
```



```
# un exemple avec une série temporelle

data(arrival)
w <- orthobasis.circ(24)
orthogram(arrival$hours, w)
class: krandtest
test number:   4
permutation number:   999
  test   obs    P(X<=obs) P(X>=obs)
1 R2Max 0.738 1          0.001
2 SkR2k 4.426 0.001      1
3 Dmax  0.651 1          0.001
4 SCE   3.53  1          0.001

par(mfrow = c(1,2))
dotcircle(arrival$hours)
dotcircle(w[,2])
par(mfrow = c(1,1))
```



R code

```
"orthogram"<- function (x, orthobas = NULL, neig = NULL, phylog = NULL,
    nrepet = 999, posinega = 0, na.action = c("fail", "mean"),
    cdot = 1.5, cfont.main = 1.5, lwd = 2, nclass, high.scores = 0)
{
    "orthoneig" <- function (obj) {
        tol <- 1e-07
        if (!inherits(obj, "neig"))
            stop("Object of class 'neig' expected")
        b0 <- neig.util.LtoG(obj)
        deg <- attr(obj, "degrees")
        m <- sum(deg)
        n <- length(deg)
        b0 <- -b0/m + diag(deg)/m
        # b0 est la matrice D-P
        eig <- eigen (b0, sym = TRUE)
        w0 <- abs(eig$values)/max(abs(eig$values))
        w0 <- which(w0<tol)
        if (length(w0)==0) stop ("abnormal output : no null eigenvalue")
        if (length(w0)==1) w0 <- (1:n)[-w0]
        else if (length(w0)>1) {
            # on ajoute le vecteur dérivé de 1n
            w <- cbind(rep(1,n),eig$vectors[,w0])
            # on orthonormalise l'ensemble
            w <- qr.Q(qr(w))
            # on met les valeurs propres à 0
            eig$values[w0] <- 0
            # on remplace les vecteurs du noyau par une base orthonormée contenant
            # en première position le parasite
            eig$vectors[,w0] <- w[,-ncol(w)]
            # on enlève la position du parasite
            w0 <- (1:n)[-w0[1]]
        }
        w0 <- rev(w0)
        rank <- length(w0)
        values <- n-eig$values[w0]*n
        eig <- eig$vectors[,w0]*sqrt(n)
        eig <- data.frame(eig)
        row.names(eig) <- names(deg)
        names(eig) <- paste("V",1:rank,sep="")
        attr(eig,"values")<-values
        eig
    }

    if (!is.numeric(x)) stop("x is not numeric")
    nobs <- length(x)
    if (!is.null(orthobas)) cas <- "orthobas"
```

```
    else if (!is.null(neig)) {
        cas <- "neig"
        orthobas <- orthoneig(neig)
    } else if (!is.null(phylog)) {
         if  (!inherits(phylog,  "phylog"))  stop  ("'phylog'  expected  with  class
'phylog'")
         orthobas <- phylog$Bscores
    } else stop ("'orthobas','neig','phylog' all NULL")

    if (!inherits(orthobas, "data.frame")) stop ("'orthobas' is not a data.frame")
    if (nrow(orthobas) != nobs) stop ("non convenient dimensions")
    if      (ncol(orthobas)      !=      (nobs-1))      stop      (paste("'orthobas'
has",ncol(orthobas),"columns, expected:",nobs-1))
    vecpro <- as.matrix(orthobas)
    npro <- ncol(vecpro)
    if (any(is.na(x))) {
        if (na.action == "fail")
            stop("missing value in 'x'")
        else if (na.action == "mean")
            x[is.na(x)] <- mean(na.omit(x))
        else stop("unknown method for 'na.action'")
    }
    w <- t(vecpro/nobs)%*%vecpro
    if (any(abs(diag(w)-1)>1e-07)) {
        # print(abs(diag(w)-1))
        stop("'orthobas' is not orthonormal for uniform weighting")
    }
    diag(w) <- 0
    if  (  any(  abs(as.numeric(w))>1e-07)  )  stop("'orthobas'  is  not  orthogonal  for
uniform weighting")
    if (nrepet < 99) nrepet <- 99
    if (posinega !=0) {
        if (posinega >= nobs-1) stop ("Non convenient value in 'posinega'")
        if (posinega <0) stop ("Non convenient value in 'posinega'")
    }

    # préparation d'un graphique à 6 fenêtres
    # 1 pgram
    # 2 pgram cumulé
    # 3-6 Tests de randomisation
    def.par <- par(no.readonly = TRUE)
    on.exit(par(def.par))
    layout (matrix(c(1,1,2,2,1,1,2,2,3,4,5,6),4,3))
    mar.old <- par("mar")
    par(mar = c(0.1, 0.1, 0.1, 0.1))
    par("usr"=c(0,1,-0.05,1))
    # layout.show(6)

    z <- x - mean(x)
    et <- sqrt(mean(z * z))
    if ( et <= tol*(max(z)-min(z))) stop ("No variance")
    z <- z/et
    sig50 <- (1:npro)/npro
    w <- .C("VarianceDecompInOrthoBasis",
        param = as.integer(c(nobs,npro,nrepet,posinega)),
        observed = as.double(z),
        vecpro = as.double(vecpro),
        phylogram = double(npro),
        phylo95 = double(npro),
        sig025 = double(npro),
        sig975 = double(npro),
        R2Max = double(nrepet+1),
        SkR2k = double(nrepet+1),
        Dmax = double(nrepet+1),
        SCE = double(nrepet+1),
        ratio = double(nrepet+1),
        PACKAGE="ade4"
    )
```

```
   ylim <- max(c(w$phylogram, w$phylo95))
   z0 <- apply(vecpro, 2, function(x) sum(z * x))
   names(w$phylogram) <- as.character(1:npro)
   phylocum <- cumsum(w$phylogram)
   lwd0=2
   fun <- function (y, last=FALSE) {
        delta <- (mp[2]-mp[1])/3
        sel <- 1:(npro - 1)
        segments(mp[sel]-delta,y[sel],mp[sel]+delta, y[sel],lwd=lwd0)
        if(last) segments(mp[npro]-delta,y[npro],mp[npro]+delta, y[npro],lwd=lwd0)
    }
    y0 <- phylocum - sig50
    h.obs <- max(y0)
    x0 <- min(which(y0 == h.obs))
    par(mar = c(3.1, 2.5, 2.1, 2.1))
    mp <- barplot(w$phylogram, col = grey(1 - 0.3 * (sign(z0) > 0)),
            ylim = c(0, ylim * 1.05))
    scores.order            <-            (1:length(w$phylogram))[order(w$phylogram,
decreasing=TRUE)[1:high.scores]]
    fun(w$phylo95,TRUE)
    abline(h = 1/npro)
    if (posinega!=0) {
        verti = (mp[posinega]+mp[posinega+1])/2
        abline (v=verti, col="red",lwd=1.5)
    }
    title(main = "Variance decomposition",font.main=1, cex.main=cfont.main)
    box()
    obs0 <- rep(0, npro)
    names(obs0) <- as.character(1:npro)
    barplot(obs0, ylim = c(-0.05, 1.05))
    abline(h=0,col="white")
    if (posinega!=0) {
        verti = (mp[posinega]+mp[posinega+1])/2
        abline (v=verti, col="red",lwd=1.5)
    }

    title(main = "Cumulative decomposition",font.main=1, cex.main=cfont.main)
    points(mp, phylocum, pch = 21, cex = cdot, type = "b")
    segments(mp[1], 1/npro, mp[npro], 1, lty = 1)
    fun(w$sig975)
    fun(w$sig025)
    arrows(mp[x0], sig50[x0], mp[x0], phylocum[x0], ang = 15, le = 0.15,
           lwd = 2)
    box()
    if (missing(nclass)) {
        nclass <- as.integer (nrepet/25)
        nclass <- min(c(nclass,40))
    }
    plot.randtest  (as.randtest  (w$R2Max[-1],w$R2Max[1],call=match.call()),main  =
"R2Max",nclass=nclass)
    if (posinega !=0) {
        plot.randtest  (as.randtest  (w$ratio[-1],w$ratio[1],call=match.call()),main
= "Ratio",nclass=nclass)
    } else {
        plot.randtest  (as.randtest  (w$SkR2k[-1],w$SkR2k[1],call=match.call()),main
= "SkR2k",nclass=nclass)
    }
    plot.randtest   (as.randtest   (w$Dmax[-1],w$Dmax[1],call=match.call()),main   =
"DMax",nclass=nclass)
    plot.randtest    (as.randtest    (w$SCE[-1],w$SCE[1],call=match.call()),main    =
"SCE",nclass=nclass)

    w$param <- w$observed <- w$vecpro <- NULL
    w$phylogram <- NULL
    w$phylo95 <- w$sig025 <- w$sig975 <- NULL
    if (posinega==0) w$ratio <- NULL
    attr(w,"call") <- match.call()
    attr(w,"class") <- "krandtest"
```

```
    if (high.scores != 0)
        return(w, scores.order)

        else
            return(w)
}

####   ####   ####   ####   la fonction orthogram(…) fait appel à une fonction en C
####   ####   ####   ####

void VarianceDecompInOrthoBasis (int *param, double *z, double *matvp,
    double *phylogram, double *phylo95,double *sig025, double *sig975,
    double *R2Max, double *SkR2k, double*Dmax, double *SCE, double *ratio)
{

    /* param contient 4 entiers : nobs le nombre de points, npro le nombre de
vecteurs
    nrepet le nombre de permutations, posinega la nombre de vecteurs de la classe
posi
    qui est nul si cette notion n'existe pas. Exemple : la base Bscores d'une
phylogénie a posinega = 0
    mais la base Ascores a posinega à prendre dans Adim
    z est un vecteur à nobs composantes de norme 1
    pour la pondération uniforme. matvp est une matrice nobsxnpro contenant en
    colonnes des vecteurs orthonormés pour la pondération uniforme. En géné
    La procédure placera
        dans phylogram les R2 de la décomposition de z dans la base matvp
        dans phylo95 les quantiles 0.95 des R2
        dans sig025 les quantiles 0.025 des R2 cumulés
        dans sig975 les quantiles 0.975 des R2 cumulés

    Ecrit à l'origine pour les phylogénies
    peut servir pour une base de vecteurs propres de voisinage */


    /* Declarations des variables C locales */
    int nobs, npro, nrepet, i, j, k, n1, n2, n3, n4;
    int irepet, posinega, *numero, *vecrepet;
    double **vecpro, *zperm, *znorm;
    double *locphylogram, *modelnul;
    double a1, provi, **simul, *copivec, *copicol;

   /* Allocation memoire pour les variables C locales */
    nobs = param[0];
    npro = param [1];
    nrepet = param [2];
    posinega = param[3];
    vecalloc (&znorm, nobs);
    vecalloc (&zperm, nobs);
    vecalloc (&copivec, npro);
    vecalloc (&copicol, nrepet);
    taballoc (&vecpro, nobs, npro);
    taballoc (&simul, nrepet, npro);
    vecalloc (&locphylogram, npro);
    vecalloc (&modelnul, npro);
    vecintalloc (&numero, nobs);
    vecintalloc (&vecrepet, nrepet);

    /* Définitions des variables C locales */
    for (i = 1 ; i<= nobs; i++) znorm[i] = z[i-1];
    for (i = 1 ; i<= npro; i++) modelnul[i] = (double) i/ (double) npro;
    k = 0;
    for (j=1; j<=npro; j++) {
        for (i=1; i<=nobs; i++) {
            vecpro[i][j] = matvp[k] ;
             k = k+1 ;
       }
```

```
   }

  /* calcul du phylogramme observé */
   for (j = 1; j<= npro; j++) {
       provi = 0;
       for (i=1; i<=nobs; i++)  provi = provi + vecpro[i][j]*znorm[i];
       provi = provi*provi/nobs/nobs;
       locphylogram[j] = provi;
  }
   for (i =1 ; i<= npro ; i++) phylogram[i-1] = locphylogram[i];
   /* calcul des simulations
   Chaque ligne de simul est un phylogramme après permutation des données */

   for (irepet=1; irepet<=nrepet; irepet++) {
       getpermutation (numero, irepet);
       vecpermut (znorm, numero, zperm);
       provi = 0;
       for (j = 1; j<= npro; j++) {
           provi = 0;
           for (i=1; i<=nobs; i++)  provi = provi + vecpro[i][j]*zperm[i];
           provi = provi*provi/nobs/nobs;
           simul[irepet][j] = provi;
       }
   }
   /* calcul du test sur le max du phylogramme */
   for (irepet=1; irepet<=nrepet; irepet++) {
        for (j=1; j<=npro; j++) copivec[j] = simul[irepet][j];
        R2Max[irepet] = maxvec(copivec);
        provi=0;
        for (j=1; j<=npro; j++) provi = provi + j*simul[irepet][j];
        SkR2k[irepet] =provi;
        if (posinega>0) {
           provi=0;
           for (j=1; j<posinega; j++) provi = provi + simul[irepet][j];
           ratio[irepet] = provi;
       }

   }
   R2Max[0] = maxvec(locphylogram);
   provi=0;
   for (j=1; j<=npro; j++) provi = provi + j*locphylogram[j];
   SkR2k[0] =provi;
   if (posinega>0) {
           provi=0;
           for (j=1; j<posinega; j++) provi = provi + locphylogram[j];
           ratio[0] = provi;
  }
  /* quantiles 95 du sup */
   n1 = (int) floor (nrepet*0.95);
   n2 = (int) ceil (nrepet*0.95);
   for (i =1; i<=npro; i++) {
       for (irepet = 1; irepet<= nrepet; irepet++) {
           copicol[irepet] = simul [irepet][i];
       }
       trirap (copicol, vecrepet);
           phylo95[i-1] = 0.5*(copicol[n1]+copicol[n2]);
  }


 for (irepet=1; irepet<=nrepet; irepet++) {
       provi = 0;
       for (j=1; j<=npro; j++) {
           provi = provi + simul[irepet][j];
           copivec[j] = provi;
       }
       for (j=1; j<=npro; j++) simul[irepet][j] = copivec[j];
   }
   n1 = (int) floor (nrepet*0.025);
```

```
    n2 = (int) ceil (nrepet*0.025);
    n3 = (int) floor (nrepet*0.975);
    n4 = (int) ceil (nrepet*0.975);
    /* quantiles 2.5 du cumul */
    for (i =1; i<=npro; i++) {
        for (irepet = 1; irepet<= nrepet; irepet++) {
            copicol[irepet] = simul [irepet][i];
        }
        trirap (copicol, vecrepet);
        sig025[i-1] = 0.5*(copicol[n1]+copicol[n2]);
        sig975[i-1] = 0.5*(copicol[n3]+copicol[n4]);
   }

    provi = 0;
    for (j=1; j<=npro; j++) {
        a1 = modelnul[j];
        provi = provi + locphylogram[j];
        locphylogram[j] = provi-a1;
        for (irepet = 1; irepet<= nrepet; irepet++) {
            simul [irepet][j] = simul [irepet][j]-a1;
        }
    }
    /* simul contient maintenant les cumulés simulés en écarts */
    /* locphylogram contient maintenant les cumulés observés en écart*/
    /* Dmax */
    for (j=1; j<=npro; j++) {
        for (irepet=1; irepet<=nrepet; irepet++) {
            for (j=1; j<=npro; j++) copivec[j] = simul[irepet][j];
            Dmax[irepet] = maxvec(copivec);
            provi=0;
            for (j=1; j<=npro; j++) provi = provi + copivec[j]* copivec[j];
            SCE[irepet] =provi;
        }
    }
    Dmax[0] = maxvec (locphylogram);
    provi=0;
    for (j=1; j<=npro; j++) provi = provi +locphylogram[j]*locphylogram[j];
    SCE[0] =provi;

    /* retour */

    freevec (znorm);
    freevec (modelnul);
    freevec(copivec);
    freevec(copicol);
    freevec (zperm);
    freetab (vecpro);
    freetab (simul);
    freevec (locphylogram);
    freeintvec (numero);
    freeintvec (vecrepet);
}
```

17. La méthode des contrastes de Felsenstein

alias

phylog2pic

description

'phylog2pic' calcule la matrice des contrastes de Felsenstein associée à une phylogénie résolue.

usage

phylog2pic(phy)

arguments

- *phy* : est un objet de la classe 'phylog'

valeurs

'phylog2pic' retourne une liste à trois composantes :

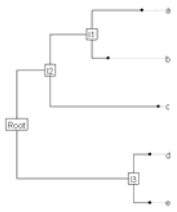
- *contrastes* : une matrice à n ligne et n-1 colonnes correspondant aux contrastes.
- *prediction* : une matrice à n ligne et n-1 colonnes permettant de calculer la valeur des prédictions sou l'hypothèse du mouvement brownien à chaque nœud.
- *variance* : un vecteur correspondant à la variance des variables aléatoires associées aux contrastes.

références

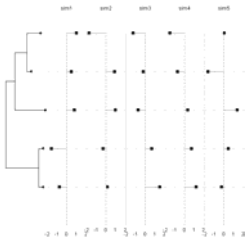
Felsenstein, J. (1985) Phylogenies and the comparative method. The American Naturalist, 125, 1-15.

exemples

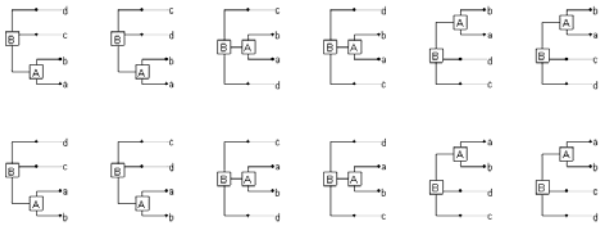
```
tre <- c("(((a:0.3,b:0.1)I1",
         ":0.25,c:0.65)",
         "I2:0.2,(d:0.1,e:0.1)",
         "I3:0.7)Root;")
phy <- newick2phylog(tre)
plot(phy, clabel.nodes = 1)
```



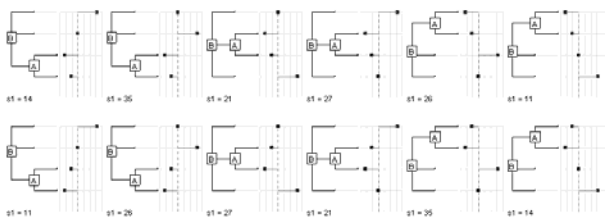
```
library(mvtnorm)
sim <- rmvnorm(5, rep(0, 5),
 sigma = phy$Wmat)
sim <- t(sim)
sim <- as.data.frame(sim)
row.names(sim) <- letters[1:5]
names(sim) <- paste(rep("sim", 5),
 1:5, sep = "")
dotchart.phylog(phy, sim,
 ceti = 0.75, csub = 0.75)
```



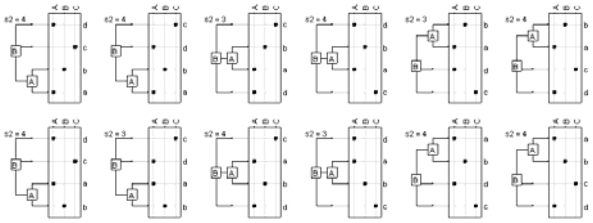
```
tre <- "((a,b)A,c,d)B;"
phy <- newick2phylog(tre)
enum <- enum.phylog(phy)
par(mfrow = c(2,6))
for (i in 1:12)
    plot.phylog(phy, y = enum[i,],
 clabel.nodes = 2, clabel.leaves = 2,
 cleaves = 1.5)
```



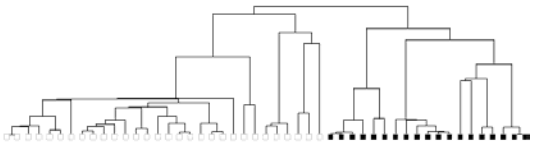
```
x <- c(-1, -2, 0, 3)
x <- as.data.frame(x)
s1 <- c(14, 35, 21, 27, 26,
 11, 11, 26, 27, 21, 35, 14)
s1 <- paste(rep("s1 = ", 12),
 s1, sep = "")
par(mfrow = c(2,6))
for(i in 1:12){
      dotchart.phylog(phy, x,
       y = enum[i,], ceti = 0, csub = 0,
       clabel.nodes = 1.5)
```

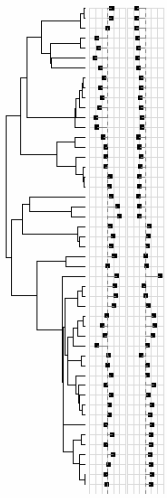


```
      scatterutil.sub(s1[i], csub = 1.5)
}

x <- as.factor(c("A", "B", "C", "A"))
x <- as.data.frame(x)
row.names(x) <- letters[1:4]
s2 <- c(4, 4, 3, 4, 3, 4,
 4, 3, 4, 3, 4, 4)
s2 <- paste(rep("s2 = ", 12),
 s2, sep = "")
for(i in 1:12){
      table.phylog(x, phy, y = enum[i,],
       labels.col = LETTERS[1:3], csize = 2,
       clabel.nod = 1.5, cleg = 0,
      clabel.row = 1.5, clabel.col = 1.5)
      scatterutil.sub(s2[i], csub = 1.5,
       possub = "topleft")
}
```

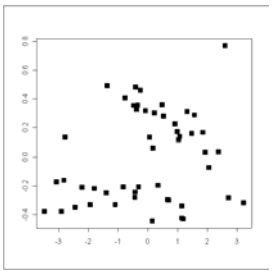


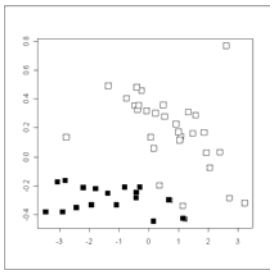
```
phy <- newick2phylog(carniherbi49$tre2)
logbodymass <- scalewt(log(
 carniherbi49$tab2$bodymass),
 scale = F)
X <- logbodymass
mtfratio <- scalewt(
 carniherbi49$tab2$mtfratio,
 scale = F)
Y <- mtfratio
clade <- c(rep(1, 19), rep(-1, 30))
tab <- cbind.data.frame(X, Y)
symbols.phylog(phy, l.squares = clade, csi = 0.35)
dotchart.phylog(phy, tab, ranging = TRUE,
 ceti = 0, csub = 0, cdot = 0.75,
 cleaves = 0, clabel.nodes = 0, f.phylog = 0.5)
```



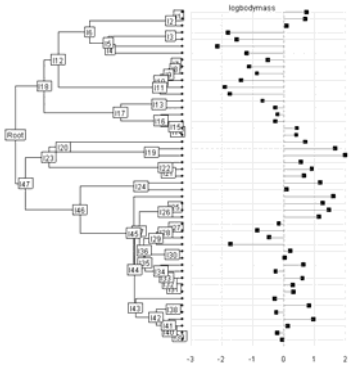


```
plot(tab, type = "n",
 xlab = "", ylab = "")
s.label(tab, add.plot = TRUE,
 clab = 0, pch = 15,
 cpoint = 1.6)

plot(tab, type = "n",
 xlab = "", ylab = "")
s.value(tab, clade,
 add.plot = TRUE, cleg = 0,
 csi = 0.4)
```

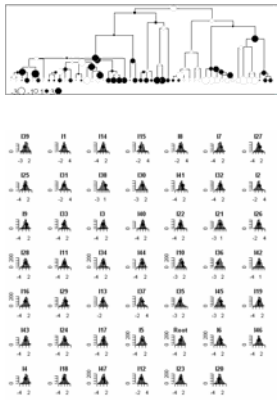




```
pic <- phylog2pic(phy)

y <- as.data.frame(logbodymass)
names(y) <- "logbodymass"
dotchart.phylog(phy, y,
 clabel.nodes = 0.75,
 csub = 0.75, ceti = 0.75,
 cleaves = 0.5)

scores.logbodymass <- t(
 pic$contrastes)%*%logbodymass
scores.logbodymass <- scores.logbodymass/
 sqrt(pic$variance)

sim <- rmvnorm(1000, rep(0, 49),
 phy$Wmat)
sim <- t(sim)
scores.sim <- t(pic$contrastes)%*%sim
scores.sim <- scores.sim/sqrt(pic$variance)
```



```
apply(scores.sim, 1, var)
   I39     I1    I14    I15     I8     I7    I27    I25    I31    I38    I30
0.9703 1.0967 1.0541 1.0393 0.9910 1.0198 0.9771 1.0271 0.9439 1.0291 0.9098
   I41    I32     I2     I9    I33     I3    I40    I22    I21    I26    I28
1.0797 1.0212 0.9806 0.9938 0.9880 1.0194 1.0275 1.0582 0.9303 0.9024 0.9793
   I11    I34    I44    I10    I36    I42    I16    I29    I13    I37    I35
1.1145 0.9957 1.0451 0.9804 0.9330 0.9734 1.0040 1.0966 0.9955 1.0405 1.0056
   I45    I19    I43    I24    I17     I5   Root     I6    I46     I4    I18
0.9802 0.9916 0.8951 0.9997 1.0359 1.0310 0.9905 0.9942 1.0335 0.9418 0.9817
   I47    I12    I23    I20
0.9865 0.9832 0.9882 0.9552

symbols.phylog(phy,
 l.circles = logbodymass,
 n.circles = as.data.frame(
 t(scores.logbodymass)),
 csize = 0.6)

u <- rbind.data.frame(
 t(scores.logbodymass),
 t(scores.sim))
names(u) <- row.names(scores.sim)
class(u) <- "krandtest"
plot(u)
```



R code

```
"phylog2pic" <- function(phylog){
    if (!inherits(phylog, "phylog"))
         stop("Non convenient data")
n <- length(phylog$nodes)
nodes <- names(phylog$nodes)
l <- length(phylog$leaves)
leaves <- names(phylog$leaves)
both <- c(leaves, nodes)

    if (l != (n + 1))
        stop("The tree must be resolved")
u <- unlist(lapply(phylog$path, function(x) x[length(x)-1]))
u <- u[duplicated(u)]
res.length <- c(phylog$leaves, phylog$nodes)
res.prediction <- cbind(diag(rep(1, l)), matrix(0, nrow = l, ncol = n))
res.variance <- rep(0, n)
res.contrastes <- as.data.frame(matrix(0, nrow = l, ncol = n))

for (i in 1:n){
    k <- (1:n)[nodes == u[i]]
    couple <- phylog$parts[[k]]
    v1 <- res.length[(1:(l + n))[both == couple[1]]]
    v2 <- res.length[(1:(l + n))[both == couple[2]]]
    res.length[l + k] <- phylog$nodes[k] + v1 * v2 / (v1 + v2)
    res.variance[k] <- v1 + v2
    res.prediction[,l + k] <- res.prediction[,(1:(l + n))[both == couple[1]]] * v2
+ res.prediction[,(1:(l + n))[both == couple[2]]] * v1
    res.prediction[,l + k] <- res.prediction[,l + k] / res.variance[k]
    res.contrastes[,k]  <-  res.prediction[,(1:(l  +  n))[both  ==  couple[1]]]  -
res.prediction[,(1:(l + n))[both == couple[2]]]
    }
res.length <- res.length[(l + 1):(l + n)]
res.prediction <- as.data.frame(res.prediction[,(l + 1):(l + n)])
row.names(res.contrastes) <- row.names(res.prediction) <- leaves
names(res.prediction) <- names(res.contrastes) <- names(res.variance) <- nodes
v <- order(res.variance)
res.contrastes <- res.contrastes[,v]
res.prediction <- res.prediction[,v]
res.variance <- res.variance[v]
return(res  <-  list(contrastes  =  res.contrastes,  prediction  =  res.prediction,
variance = res.variance))
}
```

18. Le corrélogramme de Gittleman

alias

phylog.corelogram

description

'phylog.correlogram' calcule le corrélogramme de Gittleman

usage

```
phylog.correlogram(x, phy, cutdistance = max(dist2mat(phy$Wdist^2)))
```

arguments

- *x* : est un vecteur dont on souhaite calculer le corrélogramme
- *phy* : est un objet de la classe 'phylog'
- *cutdistance* : est un vecteur donnant les distances seuils

détails

Le calcul est un peu différent de celui opéré par Gittleman et Kot (1990). On utilise l'inverse de la matrice des distances comme matrice de proximité **W** nécessaire au calcul des indices de Moran du corrélogramme. Pour chaque valeur seuil dk, on calcule la matrice de proximité correspondante **W**k en affectant la valeur 0 à tous les termes de **W** qui sont supérieurs à 1/dk.

valeurs

'phylog.correlogram' retourne les valeurs du corrélogramme ainsi sa représentation graphique.

références

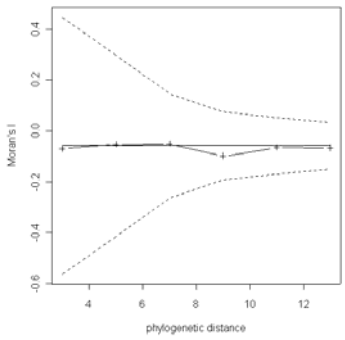
Gittleman, J.L. & Kot, M. (1990) Adaptation: statistics and a null model for estimating phylogenetic effects. Systematic Zoology, 39, 227-241.

voir également

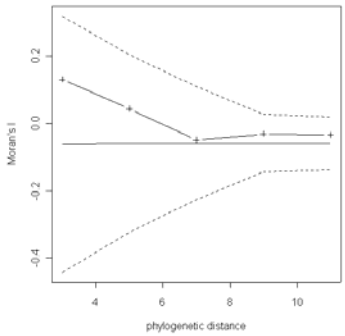
orthogram

exemples

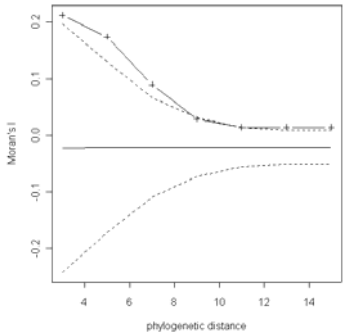
```
# absence de structure
procella.phy <- newick2phylog(procella.phy$tre)
procella.trait<-procella$traits$ALE
procella.trait<-procella$traits$ALE[
 !is.na(procella.trait)]
procella.trait <- sqrt(procella.trait)
par(mar = c(5,4,4,4))
phylog.correlogram(procella.trait,
 procella.phy, c(3,5,7,9,11,13))
[1] -0.07114619 -0.05412259 -0.05139823
 -0.09924549 -0.06616961 -0.06818362
```




```
# structure à un seul niveau :
# le corrélogramme est planté
ung.phy <- newick2phylog(ungulates$tre)
afbw <- log(ungulates$tab$afbw)
par(mar = c(5,4,4,4))
phylog.correlogram(afbw, ung.phy,
 c(3,5,7,9,11))
[1]  0.12941781  0.04246236 -0.05012657
 -0.03143698 -0.03476624
```




```
# structure diffuse
mjrochet.phy <- newick2phylog(mjrochet$tre)
tab <- log((mjrochet$tab))
tab0 <- data.frame(scalewt(tab))
par(mar = c(5,4,4,4))
phylog.correlogram(tab0[,1], mjrochet.phy,
 c(3,5,7,9,11,13,15))
[1] 0.21211944 0.17389938 0.08930495 0.02869718
 0.01410124 0.01407576 0.01407576
```



R code

```
"phylog.correlogram" <- function(x,phy,cutdistance = max(dist2mat(phy$Wdist**2))){

    require(spdep)

n <- length(phy$leaves)
d <- dist2mat(phy$Wdist**2)
diag(d) <- rep(1, n)    ## pour que l'inverse de la distance soit toujours défini
res <- rep(0, length(cutdistance))
resmean <- res
res025 <- res
res975 <- res
pvalue <- res
for(i in 1:length(cutdistance)){

# on calcul W
w <- d
dFALSE <- d > cutdistance[i]
dTRUE <- d <= cutdistance[i]
w[dFALSE] <- rep(0, sum(dFALSE))
w[dTRUE] <- 1/(d[dTRUE])
diag(w) <- rep(0, n)
w.w <- mat2listw(w)
u <- moran.test(x, w.w, randomisation = FALSE, zero.policy = TRUE, alternative =
"two.sided")
pvalue[i] <- u$p.value
res[i] <- u$estimate[1]
resmean[i] <- u$estimate[2]
res025[i] <- resmean[i] - 1.96*sqrt(u$estimate[3])
res975[i] <- resmean[i] + 1.96*sqrt(u$estimate[3])

}

# sortie graphique
u <- range(res, resmean, res025, res975)
plot(cutdistance, res, type = "b", pch = 3, ylim = u, xlab = "phylogenetic
distance", ylab = "Moran's I")
lines(cutdistance, res025, lty = 2)
lines(cutdistance, res975, lty = 2)
lines(cutdistance, resmean, lty = 1)
return(res)
}
```

19. Classe d'objet pour l'utilisation des phylogénies

alias

phylog, print.phylog, phylog.extract, phylog.permut

description

'newick2phylog', 'taxo2phylog', 'hclust2phylog' créent des objets de la classe 'phylog'. Un objet de la classe 'phylog' est une liste décrivant une arborescence et contient des éléments pour faire l'analyse d'un trait biologique en face de cette structure.

- 'phylog.extract' extrait de la phylogénie le sous-arbre des descendants d'un nœud donné.
- 'phylog.permut' permute les nœuds ou les feuilles de la phylogénie dans toute configuration compatible.

usage

```
print.phylog( (x, ...)
phylog.extract (phylog,node,distance=T)
phylog.permut (phylog, list.nodes = lapply(phylog$parts,
            function(a) if (length(a) == 1) a else sample(a)), distance = TRUE)
```

arguments

- *phylog,x* : est un objet de la classe 'phylog'
- *node* : est une chaîne de caractères donnant le nom d'un noeud
- *distance* : est une variable binaire. Si elle prend la valeur TRUE, l'extraction conserve les distances, si FALSE toutes les distance sont mises à l'unité.
- *list.nodes* :est une liste de longueur arbitraire dont les noms des éléments sont des nœuds de 'phylog' et dont les contenus des éléments sont des permutations des contenus des éléments correspondants de phylog$parts

détails

Un objet de la classe 'phylog' est une liste comprenant :

- **tre** : une chaîne de caractères contenant un arbre au format 'Newick' entièrement labellisé et non valué
- **leaves** : un vecteur dont les éléments sont les feuilles de l'arbre et les valeurs sont les longueurs de branches qui y mènent
- **nodes** : un vecteur dont les éléments sont les nœuds de l'arbre et les valeurs sont les longueurs de branches qui y mènent
- **parts** : une liste donnant pour chaque nœud de l'arbre le vecteur des noms des descendants
- **paths** : une liste donnant pour chaque nœud ou feuille de l'arbre le chemin de la racine à l'élément sous forme d'un vecteur de noms
- **droot** : un vecteur donnant pour chaque nœud ou feuille de l'arbre la distance à la racine
- **call** : l'ordre d'appel

Elle contient en outre, en option, les outils statistiques utilisés dans diverses fonctions :

- **Wmat** : la matrice des longueurs des chemins au premier ancêtre commun.
- **Wdist** : l'objet de la classe 'dist' contenant la matrice des racines carrées des distances nodales.
- **AWvalues** : valeurs propres de Wmat
- **Wscores** : vecteurs propres de Wmat
- **Amat** : la matrice des inverses des produits des degrés des nœuds du plus court chemin, dite matrice de proximité topologique
- **Avalues** : valeurs propres de Amat
- **Adim** : nombre de valeurs propres positives
- **Ascores** : vecteurs propres de Amat
- **Aparam** : attributs associés aux nœuds (nombre de permutations du sous-arbre enracinés à chaque nœud)

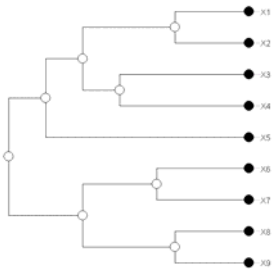
- **Bindica** : data.frame donnant pour chaque nœud les indicatrices de classes (sauf la dernière) des parties définies par ses descendants immédiats.
- **Bscores** : data.frame contenant la base orthonormée pour la pondération uniforme issue de l'orthonormalisation de 'Bindica'. Si f est le nombre de feuilles, ce data frame contient en colonnes f-1 scores sur les feuilles de moyenne 0, variance 1 et covariances nulles 2 à 2 pour la pondération uniforme.
- **Blabels** : vecteur de chaines de caractères donnant pour chaque nœud le noms des colonnes de 'scores' qui le concerne(utilisé dans 'phylogram.plot')

valeurs

'phylog.extract' et 'phylog.permut' renvoient des objets de la classe 'phylog'.

exemples

```
marthans.tre <- c( "((((1:4,2:4)a:5,",
                    "(3:7,4:7)b:2)c:2,",
                     "5:11)d:2,((6:5,7:5)e:4"
                     ",(8:4,9:4)f:5)g:4);")
marthans.phy <- newick2phylog(marthans.tre)
plot.phylog(marthans.phy, cnode = 3,
 f = 0.8, cle = 3)
```



```
unclass(marthans.phy)
```

$tre : l'arbre non valué en format newick

```
[1] "((((X1,X2)a,(X3,X4)b)c,X5)d,((X6,X7)e,(X8,X9)f)g)Root;"
```

$leaves : la liste des feuilles et leurs distances à leur premier ancêtre (longueur de l'arête adjacente supérieure)

```
X1 X2 X3 X4 X5 X6 X7 X8 X9
 4  4  7  7 11  5  5  4  4
```

$nodes : la liste des nœuds et leurs distances à leur premier ancêtre (longueur de l'arête adjacente supérieure)

```
   a    b    c    d    e    f    g Root
   5    2    2    2    4    5    4    0
```

$parts : la liste des divisions : chaque élément est un noeud

```
$parts$a
[1] "X1" "X2"
$parts$b
[1] "X3" "X4"
...
$parts$g
[1] "e" "f"
$parts$Root
[1] "d" "g"
```

$paths : la liste des chemins descendants : chaque élément est une feuille puis un nœud

```
$paths$X1
[1] "Root" "d"    "c"    "a"    "X1"
$paths$X2
[1] "Root" "d"    "c"    "a"    "X2"
...
$paths$X9
[1] "Root" "g"    "f"    "X9"

$paths$a
[1] "Root" "d"    "c"    "a"

...

$paths$g
[1] "Root" "g"
```

```
$paths$Root
[1] "Root"
```

$droot : les distances à la racine

```
  X1   X2   X3   X4   X5   X6   X7   X8   X9    a    b    c    d    e    f    g
  13   13   13   13   13   13   13   13   13    9    6    4    2    8    9    4
Root
   0
```

$call : la commande ayant permis la création de cet objet

```
newick2phylog(x.tre = marthans.tre)
```

$Wmat : la métrique phylogénétique W

```
    1  2  3  4  5  6  7  8  9
X1 13  9  4  4  2  0  0  0  0
X2  9 13  4  4  2  0  0  0  0
X3  4  4 13  6  2  0  0  0  0
X4  4  4  6 13  2  0  0  0  0
X5  2  2  2  2 13  0  0  0  0
X6  0  0  0  0  0 13  8  4  4
X7  0  0  0  0  0  8 13  4  4
X8  0  0  0  0  0  4  4 13  9
X9  0  0  0  0  0  4  4  9 13
```

$Wdist : la matrice des distances euclidiennes associées à W

```
      X1    X2    X3    X4    X5    X6    X7    X8
X2 2.828
X3 4.243 4.243
X4 4.243 4.243 3.742
X5 4.690 4.690 4.690 4.690
X6 5.099 5.099 5.099 5.099 5.099
X7 5.099 5.099 5.099 5.099 5.099 3.162
X8 5.099 5.099 5.099 5.099 5.099 4.243 4.243
X9 5.099 5.099 5.099 5.099 5.099 4.243 4.243 2.828
```

$Wvalues : les valeurs propres de l'opérateur associé à W

```
[1] 1.0190 0.4655 0.4483 0.4120 0.2414 0.1724 0.1379 0.1379
```

$Wscores : les vecteurs propres de l'opérateur associé à W

```
        W1       W2          W3       W4          W5          W6          W7
X1 -1.0711  0.16904  1.359e+00 -0.69146  1.378e-15  1.875e-16  2.118e+00
X2 -1.0711  0.16904  1.359e+00 -0.69146 -1.191e-15 -2.656e-16 -2.118e+00
X3 -0.8979 -0.03394 -6.794e-01  1.49371 -2.121e+00 -1.428e-15  5.204e-16
X4 -0.8979 -0.03394 -6.794e-01  1.49371  2.121e+00  1.395e-15 -1.790e-15
X5 -0.4733 -0.25024 -2.038e+00 -1.88666 -1.010e-15 -7.825e-17 -3.747e-16
X6  1.0684 -1.49477  3.397e-01  0.09329 -2.227e-15  2.121e+00 -3.331e-16
X7  1.0684 -1.49477  3.397e-01  0.09329  9.369e-16 -2.121e+00  0.000e+00
X8  1.1371  1.48480  1.749e-15  0.04780  3.244e-16 -8.913e-16  1.171e-01
X9  1.1371  1.48480  1.915e-15  0.04780 -1.023e-16  8.573e-16 -1.171e-01
           W8
X1  1.171e-01
X2 -1.171e-01
X3  2.908e-16
X4 -9.257e-16
X5 -5.386e-16
X6 -4.424e-16
X7 -3.123e-16
X8 -2.118e+00
X9  2.118e+00
```

$Amat : la matrice A associée au test d'Abouheif

```
         1       2       3       4      5       6       7       8       9
X1 0.06250 0.50000 0.12500 0.12500 0.1250 0.01563 0.01563 0.01563 0.01563
X2 0.50000 0.06250 0.12500 0.12500 0.1250 0.01563 0.01563 0.01563 0.01563
X3 0.12500 0.12500 0.06250 0.50000 0.1250 0.01563 0.01563 0.01563 0.01563
X4 0.12500 0.12500 0.50000 0.06250 0.1250 0.01563 0.01563 0.01563 0.01563
X5 0.12500 0.12500 0.12500 0.12500 0.2500 0.06250 0.06250 0.06250 0.06250
```

```
X6 0.01563 0.01563 0.01563 0.01563 0.0625 0.12500 0.50000 0.12500 0.12500
X7 0.01563 0.01563 0.01563 0.01563 0.0625 0.50000 0.12500 0.12500 0.12500
X8 0.01563 0.01563 0.01563 0.01563 0.0625 0.12500 0.12500 0.12500 0.50000
X9 0.01563 0.01563 0.01563 0.01563 0.0625 0.12500 0.12500 0.50000 0.12500
```

$Avalues : les valeurs propres de l'opérateur associé à A

```
[1]  7.155  3.375  2.813  1.282 -3.375 -3.375 -3.938 -3.938
```

$Adim : le nombre de valeurs propres positives

```
[1] 4
```

$Ascores : les vecteurs propres de l'opérateur A

```
        A1          A2          A3      A4          A5          A6          A7
X1 -0.9981  3.428e-16  1.500e+00  0.5038  0.000e+00  8.389e-16  2.116e+00
X2 -0.9981  3.759e-16  1.500e+00  0.5038  0.000e+00 -4.545e-16 -2.116e+00
X3 -0.9981 -4.489e-17 -1.500e+00  0.5038  2.153e-16 -2.929e-15 -1.512e-01
X4 -0.9981 -4.668e-16 -1.500e+00  0.5038 -3.222e-16  3.235e-15  1.512e-01
X5 -0.4107  1.147e-16  6.922e-16 -2.7984 -1.310e-17  1.762e-16  0.000e+00
X6  1.1008 -1.500e+00 -2.082e-16  0.1958  2.337e-01 -2.108e+00  9.159e-16
X7  1.1008 -1.500e+00 -2.082e-16  0.1958 -2.337e-01  2.108e+00 -7.494e-16
X8  1.1008  1.500e+00 -7.910e-16  0.1958 -2.108e+00 -2.337e-01  1.665e-16
X9  1.1008  1.500e+00 -8.743e-16  0.1958  2.108e+00  2.337e-01 -8.327e-17
           A8
X1 -1.512e-01
X2  1.512e-01
X3 -2.116e+00
X4  2.116e+00
X5 -3.643e-17
X6  1.270e-15
X7 -1.233e-15
X8  1.041e-16
X9  1.041e-16
```

$Aparam : les paramètres topologiques caractérisant chaque noeud

```
     x1 x2     x3     x4
a     2  2 0.6931 0.0000
b     2  2 0.6931 0.0000
c     2  4 2.0794 0.6667
d     2  5 2.7726 0.8667
e     2  2 0.6931 0.0000
f     2  2 0.6931 0.0000
g     2  4 2.0794 0.6667
Root  2  9 5.5452 0.9993
```

$Bindica : la matrice des indicatrices de classes permettant de définir la base B

```
   d c f b X4 X7 X9 X2
X1 1 1 0 0  0  0  0  0
X2 1 1 0 0  0  0  0  1
X3 1 1 0 1  0  0  0  0
X4 1 1 0 1  1  0  0  0
X5 1 0 0 0  0  0  0  0
X6 0 0 0 0  0  0  0  0
X7 0 0 0 0  0  1  0  0
X8 0 0 1 0  0  0  0  0
X9 0 0 1 0  0  0  1  0
```

$Bscores : la base B

```
        B1          B2          B3          B4          B5          B6     B7
X1 -0.8944 -6.708e-01  1.665e-16  1.500e+00  8.536e-16 -1.667e-17  0.000
X2 -0.8944 -6.708e-01  4.163e-17  1.500e+00  1.121e-15  2.907e-17  0.000
X3 -0.8944 -6.708e-01 -6.245e-17 -1.500e+00  2.121e+00  6.396e-17  0.000
X4 -0.8944 -6.708e-01  2.909e-17 -1.500e+00 -2.121e+00 -6.435e-17  0.000
X5 -0.8944  2.683e+00  2.571e-17  7.078e-16 -6.200e-16  1.713e-18  0.000
X6  1.1180  4.163e-17  1.500e+00 -1.249e-16  4.477e-16  2.121e+00  0.000
X7  1.1180  4.163e-17  1.500e+00 -1.249e-16  1.146e-16 -2.121e+00  0.000
X8  1.1180  4.163e-17 -1.500e+00  4.163e-17 -2.184e-16  2.084e-17 -2.121
X9  1.1180  4.163e-17 -1.500e+00  4.163e-17 -2.184e-16  2.084e-17  2.121
           B8
```

```
X1 -2.121e+00
X2  2.121e+00
X3  8.327e-17
X4  3.331e-16
X5  9.159e-16
X6 -1.665e-16
X7  1.665e-16
X8 -1.665e-16
X9  1.665e-16
```

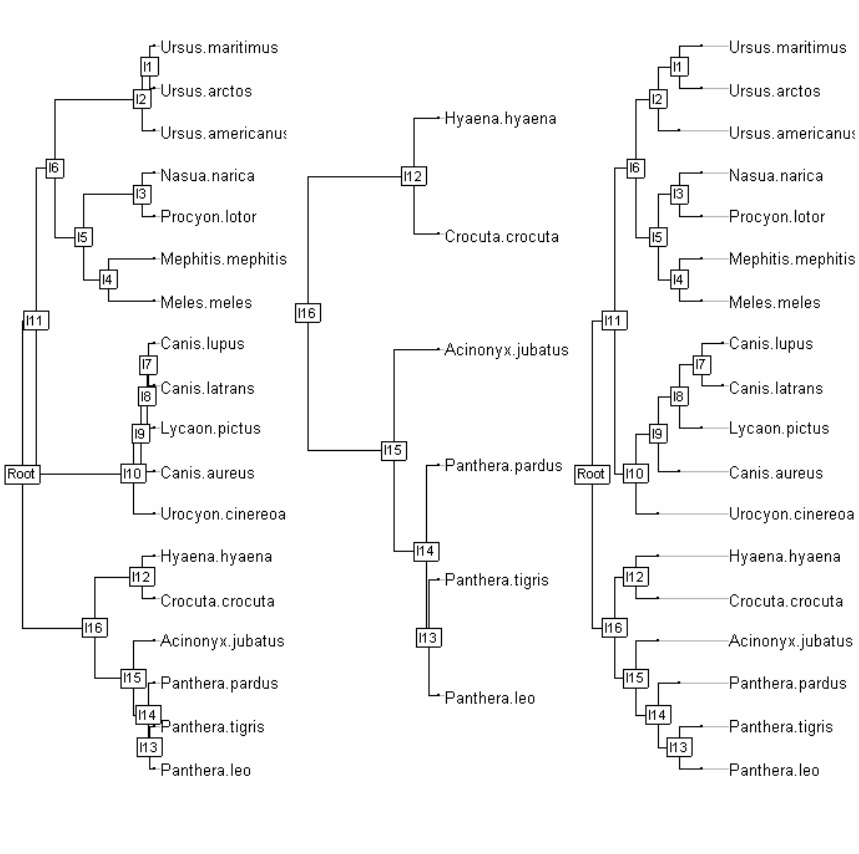
$Bvalues : la norme des vecteurs de B pour la métrique phylogénétique W

```
     d      c      f      b     X4     X7     X9     X2
64.667 57.111 30.222 25.111  9.778  9.778  9.556  9.111
```

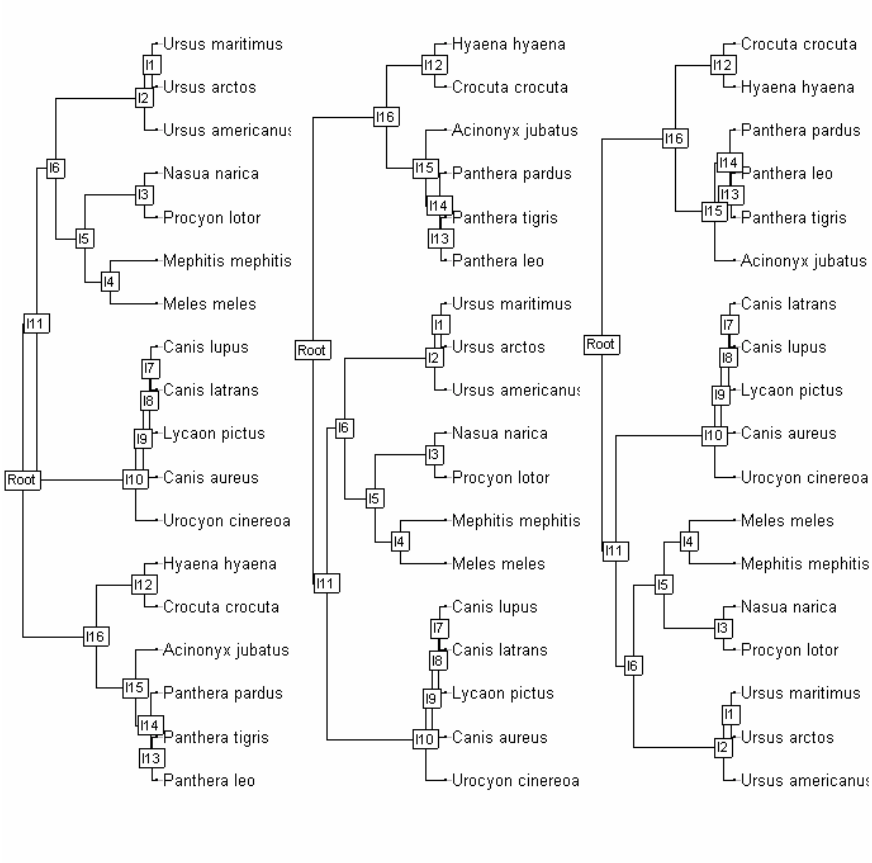
$Blabels : le nom des vecteurs associés à chaque noeud

```
   a    b    c    d    e    f    g Root
 "8"  "5"  "4"  "2"  "6"  "7"  "3"  "1"
```

```
# la fonction phylog.extract
carni18.phy <- newick2phylog(carni18$tre)
par(mfrow = c(1,3))
plot.phylog(carni18.phy, clabel.n = 1.5,clabel.l = 2)
plot.phylog(phylog.extract(carni18.phy,"I16",dist=TRUE),clabel.n=1.5,clabel.l=2)
plot.phylog(phylog.extract(carni18.phy,"Root",dist=FALSE),clabel.n=1.5,clabel.l=2)
```




```
# la fonction phylog.permut
par(mfrow=c(1,3))
plot.phylog(carni18.phy,clabel.n=1.5,clabel.l=2)
plot.phylog(phylog.permut(carni18.phy,list=list(Root=c("I16","I11"))),clabel.n=1.5,
clabel.l=2)
plot.phylog(phylog.permut(carni18.phy),clabel.n=1.5,clabel.l=2)
```



R code

```
"print.phylog" <- function (x, ...) {
    phylog <- x
    if (!inherits(phylog, "phylog"))
        stop("for 'phylog' object")
   leaves.n <- length(phylog$leaves)
    nodes.n <- length(phylog$nodes)
     cat("Phylogenetic tree with",leaves.n,"leaves and",nodes.n,"nodes\n")
    cat("$class: ")
    cat(class(phylog))
    cat("\n$call: ")
    print(phylog$call)
    cat("$tre: ")
    l0 <- nchar(phylog$tre)
    if (l0 < 50)
        cat(phylog$tre, "\n")
    else {
        cat(substring(phylog$tre, 1, 25))
        cat("...")
        cat(substring(phylog$tre, l0 - 26, l0), "\n")
    }
    cat("\n")
    n1 <-paste("$",names(phylog)[2:6],sep="")
    sumry   <-   array("   ",   c(length(n1),   3),   list(n1,   c("class",   "length",
"content")))
    # leaves
    k     <-     1;     sumry[k,1]     <-     "numeric"     ;     sumry[k,2]     <-
as.character(length(phylog$leaves))
    sumry[k,3] <- "length of the first preceeding adjacent edge"
    #nodes
    k     <-     2     ;     sumry[k,1]     <-     "numeric"     ;     sumry[k,2]     <-
as.character(length(phylog$nodes))
    sumry[k,3]  <- "length of the first preceeding adjacent edge"
    #parts
    k <-3; sumry[k,1] <- "list";sumry[k,2] <- as.character(length(phylog$parts))
    sumry[k,3]  <- "subsets of descendant nodes"
```

```
#paths
k = 4; sumry[k,1] <- "list";sumry[k,2] <- as.character(length(phylog$paths))
sumry[k,3]  <- "path from root to node or leave"
#droot
k = 5; sumry[k,1] <- "numeric";sumry[k,2] <- as.character(length(phylog$droot))
sumry[k,3]  <- "distance to root"
print.noquote(sumry)
cat("\n")
if (is.null(phylog$Wmat)) return(invisible())

n1 <- names(phylog)[-(1:7)]
n1 <- paste("$",n1,sep="")
sumry <- array(" ", c(length(n1), 3), list(n1, c("class", "dim", "content")))
# 8 Wmat
k = 1
sumry[k,1] <- "matrix"
sumry[k,2] <- paste(nrow(phylog$Wmat),ncol(phylog$Wmat),sep="-")
sumry[k,3] <- "W matrix : root to the closest ancestor"
#9 Wdist
k = 2
sumry[k,1] <- "dist" ;
sumry[k,2] <- as.character(length(phylog$Wdist))
sumry[k,3] <- "Nodal distances"
# 10 Wvalues
k = 3
sumry[k,1] <- "numeric"
sumry[k,2] <- length(phylog$Avalues)
sumry[k,3] <- "Eigen values of QWQ/sum(Q)"
#11 "Wscores"
k = 4
sumry[k,1] <- "data.frame"
sumry[k,2] <- paste(nrow(phylog$Wscores),ncol(phylog$Wscores),sep="-")
sumry[k,3] <- "Eigen vectors of QWQ '1/n' normed"
#12 "Amat"
k = 5
sumry[k,1] <- "matrix"
sumry[k,2] <- paste(nrow(phylog$Amat),ncol(phylog$Amat),sep="-")
sumry[k,3] <- "Topological proximity matrix A"
#13 Avalues
k = 6
sumry[k,1] <- "numeric"
sumry[k,2] <- length(phylog$Avalues)
sumry[k,3] <- "Eigen values of QAQ matrix"
#14 Adim
k = 7
sumry[k,1] <- "integer"
sumry[k,2] <- "1"
sumry[k,3] <- "number of positive eigen values of QAQ"
#15 Ascores
k = 8
sumry[k,1] <- "data.frame"
sumry[k,2] <- paste(nrow(phylog$Ascores),ncol(phylog$Ascores),sep="-")
sumry[k,3] <- "Eigen vectors of QAQ '1/n' normed"
#16 Aparam
k = 9
sumry[k,1] <- "data.frame"
sumry[k,2] <- paste(nrow(phylog$Aparam),ncol(phylog$Aparam),sep="-")
sumry[k,3] <- "Topological indices for nodes"
# 17 Bindica
k = 10
sumry[k,1] <- "data.frame"
sumry[k,2] <- paste(nrow(phylog$Bindica),ncol(phylog$Bindica),sep="-")
sumry[k,3] <- "class indicator from nodes"
# 18 Bscores
k = 11
sumry[k,1] <- "data.frame"
sumry[k,2] <- paste(nrow(phylog$Bscores),ncol(phylog$Bscores),sep="-")
sumry[k,3] <- "Topological orthonormal basis '1/n' normed"
```

```
    # 19 Bvalues
    # 20 Blabels
    k=12
    sumry[k,1] <- "character"
    sumry[k,2] <- length(phylog$Blabels)
    sumry[k,3] <- "Nodes labelling from orthonormal basis"
    print.noquote(sumry)
    return(invisible())
}

####   ####   ####   ####
####   ####   ####   ####

phylog.extract<-function(phylog,node,distance=TRUE){
    #extrait d'une phylogénie phylog le sous-arbre enraciné au noeud node
    #il serait intéressant de traduire cett fonction en C
    #en ne travaillant que sur les chaines de caractères newick
    tre2tre<-function(res){
        # cette fonction assure la conversion de l'objet res
        # en son équivalent munie des distances
        # on affecte les distances au noeud le plus proche pour chaque feuilles et
noeuds
        for(i in 1:length(leaves.names)) {
            res<-
sub(paste(leaves.names[i],",",sep=""),paste(leaves.names[i],":",phylog$leaves[i],",
",sep=""),res)
        }
        for(i in 1:length(leaves.names)) {
            res<-
sub(paste(leaves.names[i],")",sep=""),paste(leaves.names[i],":",phylog$leaves[i],")
",sep=""),res)
        }
        for(i in 1:length(nodes.names)) {
            res<-
sub(paste(nodes.names[i],",",sep=""),paste(nodes.names[i],":",phylog$nodes[i],",",s
ep=""),res)
        }
        for(i in 1:length(nodes.names)) {
            res<-
sub(paste(nodes.names[i],")",sep=""),paste(nodes.names[i],":",phylog$nodes[i],")",s
ep=""),res)
        }
        res
    }
    #variables locales
    add.t <- !is.null(phylog$Wmat)
    tre<-phylog$tre
    nodes.names<- names(phylog$nodes)
    leaves.names<- names(phylog$leaves)
    node.number<- grep(node, nodes.names)

    #on détermine la feuilles la plus à gauche associée au noeud
    leave<-node
    k<-0
    while(length(grep(leave,leaves.names))==0) {
        k<-k+1
        leave.number<-grep(leave, nodes.names)[1]
        leave<-phylog$parts[[leave.number]][1]
    }
    #on construit la chaine de caractère associée à l'arbre enraciné au noeud
    leave.pos<-regexpr(leave,tre)
    node.pos<-regexpr(node,tre)
    res<-substr(tre,leave.pos,node.pos-1)
    res<-paste(res,node,sep="")
    if (k==0) parentheses<-"" else parentheses<-"("
    if (k > 1) {
        for(i in 2:k){
            parentheses<-paste(parentheses,"(", sep="")
```

```
        }
    }
    res<-(paste(parentheses, res, sep=""))
    res <- paste(res,";",sep="")
    if (distance) res<-tre2tre(res)
return(res)
   res <- newick2phylog(res, add.tools= add.t,call=match.call())
   res
}

####   ####   ####   ####
####   ####   ####   ####

phylog.permut <- function(phylog,list.nodes = NULL, distance = TRUE){
    if  (is.null(list.nodes))  list.nodes  <-  lapply(phylog$parts,function(a)  if
(length(a)==1) a else sample(a))
    ############################
    adddistances<-function(){
        # cette fonction assure la conversion de tre
        # en son équivalent muni des distances
        for(i in 1:length(leaves.names)) {
             tre<<-
sub(paste(leaves.names[i],",",sep=""),paste(leaves.names[i],":",phylog$leaves[i],",
",sep=""),tre,extended=FALSE)
        }
        for(i in 1:length(leaves.names)) {
            tre<<-
sub(paste(leaves.names[i],")",sep=""),paste(leaves.names[i],":",phylog$leaves[i],")
",sep=""),tre,extended=FALSE)
        }
      for(i in 1:length(nodes.names)) {
            tre<<-
sub(paste(nodes.names[i],",",sep=""),paste(nodes.names[i],":",phylog$nodes[i],",",s
ep=""),tre,extended=FALSE)
        }
        for(i in 1:length(nodes.names)) {
            tre<<-
sub(paste(nodes.names[i],")",sep=""),paste(nodes.names[i],":",phylog$nodes[i],")",s
ep=""),tre,extended=FALSE)
        }
    }
    ############################
    extract<-function(node) {
        # extrait de tre le sous-arbre enraciné au noeud node
        # il serait intéressant de traduire cett fonction en C,
        # en ne travaillant que sur les chaines de caractères newick
        # node.number<- grep(node, nodes.names)
        # on détermine la feuille la plus à gauche associée au noeud
        # utilise la liste phylogparts contenant les descendants
        leave <- node
        k <- 0
        while(length(grep(leave,leaves.names))==0) {
            k <- k+1
            leave <- phylogparts[[leave]][1]
        }
        #on construit la chaine de caractères associée à l'arbre enraciné au noeud
        if (regexpr(paste(leave,")",sep=""),tre) == -1) {
            leave.pos <- regexpr(paste(leave,",",sep=""),tre)
        } else {
            leave.pos <- regexpr(paste(leave,")",sep=""),tre)
        }
        if (regexpr(paste(node,")",sep=""),tre) == -1) {
            node.pos <- regexpr(paste(node,",",sep=""),tre)
        } else {
            node.pos <- regexpr(paste(node,")",sep=""),tre)
        }
        res<-substr(tre,leave.pos,node.pos-1)
        res<-paste(res,node,sep="")
```

```
        if (k==0) parentheses<-"" else parentheses<-"("
        if(k > 1) {
            for(i in 2:k){
                parentheses<-paste(parentheses,"(", sep="")
            }
        }
        res<-(paste(parentheses, res, sep=""))
        return(res)
    }
    ############################
    permute <- function (node) {
        # on remplace l'ordre initial conservé dans phylogparts[[node]]
        # par l'ordre final conservé dans list.nodes[[node]]
        # phylogparts[[node]] est mis à jour à la sortie
        new.part <- list.nodes[[node]]
        if (length(new.part)==1) return(invisible())
        old.part <- phylogparts[[node]]
        if (all (old.part==new.part)) return(invisible())
        for (k in 1:(length(new.part)-1)) {
            if (old.part[k]!=new.part[k]) {
                n1 <- old.part[k]
                n2 <- new.part[k]
                u1 <- extract(n1)
                u1.pos <- regexpr(paste(u1,"[,);]",sep=""),tre,ext=FALSE)
                u1.fin <- u1.pos+attr(u1.pos,"match.length")-1
                lastcar1 <- substring(tre, u1.fin, u1.fin)
                u2 <- extract(n2)
                u2.pos<-regexpr(paste(u2,"[,);]",sep=""),tre,ext=FALSE)
                u2.fin <- u2.pos+attr(u2.pos,"match.length")-1
                lastcar2 <- substring(tre, u2.fin, u2.fin)
                tre                                                    <-
sub(paste(u1,lastcar1,sep=""),"Restunlogicielformidable",tre,extended=FALSE)
                tre                <-                sub(paste(u2,lastcar2,sep=""),
paste(u1,lastcar2,sep=""),tre,extended=FALSE)
                tre   <-   sub("Restunlogicielformidable",paste(u2,lastcar1,sep=""),
tre,extended=FALSE)
                old.part[old.part==n1] <- "1234564789"
                old.part[old.part==n2] <- n1
                old.part[old.part=="1234564789"] <- n2
             }
        }
        phylogparts[[node]] <<- new.part
    }
    ############################
    verif <- function(node) {
        new.part <- sort(list.nodes[[node]])
        old.part <- sort(phylogparts[[node]])
        if (!(all(new.part==old.part))) return (FALSE)
        return (TRUE)
    }
    if(!inherits(phylog,"phylog")) stop ("Object with class 'phylog' expected")
    nodes.names<- names(phylog$nodes)
    nodes.number<- length(nodes.names)
    leaves.names<- names(phylog$leaves)
    droot <- phylog$droot
    new.names <- names(list.nodes)
    phylogparts <- phylog$parts
    phylogleaves <- phylog$leaves
    if (any(!new.names%in%nodes.names)) stop ("Non convient name in 'list.nodes'")
    wverif <- unlist(lapply(new.names,verif))
    if (any(!wverif)) stop ("Non convient content in 'list.nodes'")
    tre <- phylog$tre
    add.t <- !is.null(phylog$Wmat)
    for (node in new.names) permute(node)
    if (distance) adddistances ()
    res <- newick2phylog(tre, add.tools= add.t, call = match.call())
    return(res)
}
```

20. Représentation graphique d'un objet de la classe 'phylog'

alias

plot.phylog, radial.phylog, enum.phylog

description

'plot.phylog' dessine des arbres en dendrogrammes.
'radial.phylog' dessine des arbres en cercles.
'enum.phylog' est une fonction pédagogique qui énumère toutes les permutations compatibles avec la topologie de la phylogénie.

usage

```
plot.phylog (x, y=NULL,
    f.phylog = 0.5, cleaves = 1, cnodes = 0,
    labels.leaves = names(x$leaves), clabel.leaves = 1,
    labels.nodes = names(x$nodes), clabel.nodes = 0,
    sub = "", csub = 1.25, possub = "bottomleft", draw.box = FALSE, . . .)

radial.phylog (phylog, circle = 1,
    cleaves = 1, cnodes = 0,
    labels.leaves = names(phylog$leaves), clabel.leaves = 1,
    labels.nodes = names(phylog$nodes), clabel.nodes = 0, draw.box = FALSE)

enum.phylog (phylog)
```

arguments

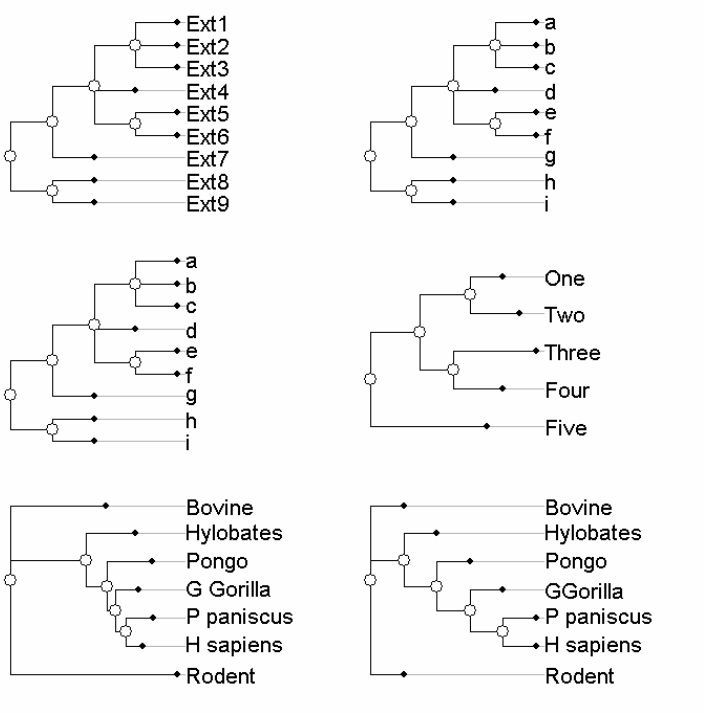
- *x, phylog* : est un objet de la classe 'phylog'
- *y* : est un vecteur donnant la position des feuilles
- *circle* : est un entier donnant la taille du cercle extérieur
- *f.phylog* : est un entier donnant la proportion de la fenêtre consacrée à l'arbre, utile pour laisser la place demandée par les noms des feuilles
- *cleaves* : est un entier donnant la taille des points représentant les feuilles (utilisé avec par("cex")*cleaves)
- *cnodes* : est un entier donnant la taille des points représentant les nœuds (utilisé avec par("cex")*cnodes)
- *label.leaves* : est un vecteur de chaînes de cractères donnant les étiquettes des feuilles (utilisé si cleaves>0)
- *clabel.leaves* : est un entier donnant la aille des caractères des étiquettes des feuilles. Si 0, il n'y a pas d'étiquettes. Utilisé avec par("cex")*clabel.leaves.
- *label.nodes* : est un vecteur de chaînes de caractères donnant les étiquettes des nœuds (utilisé si cnodes>0)
- *clabel.nodes* : est un entier donnant la taille des caractères des étiquettes des nœuds. Si 0, il n'y a pas d'étiquettes. Utilisé avec par("cex")*clabel.nodes.
- *sub* : est une chaîne de caractères sous-titrant le dessin
- *csub* : est une entier donnant la taille des caractères pour la chaîne sub (utilisé avec par(cex)*csub)
- *possub* : est une chaîne de caractères donnant la position de la chaîne sub ("topright", "topleft", "bottomright", "bottomleft")
- *draw.box* : est une variable bianire. Si TRUE, le cadre de la figure est tracé
- … : arguments supplémentaires

détails

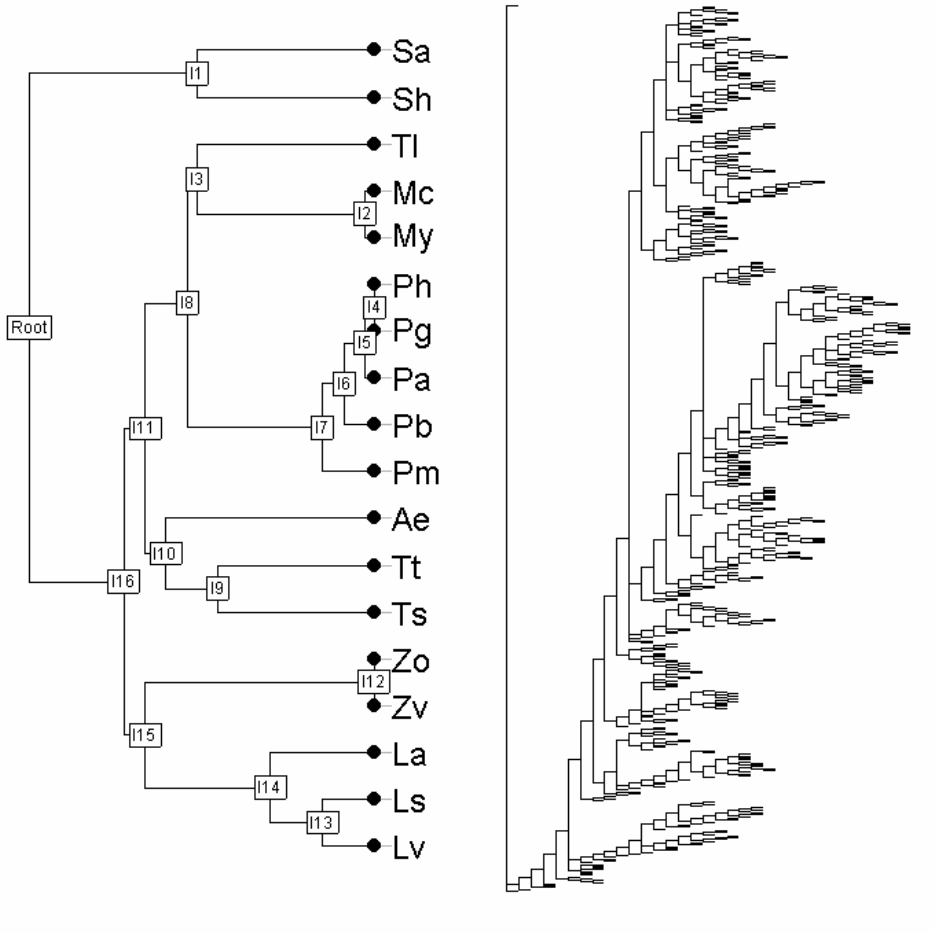
Dans plot.phylog il existe un paramètre y qui positionne les feuilles en ordonnée. Les vecteurs y acceptés sont des permutations de $\{1,2,...,n\}$ où n est le nombre de feuilles. Ces permutations doivent être compatibles avec la phylogénie. La fonction 'enum.phylog' énumère toutes les permutations possibles.

exemples

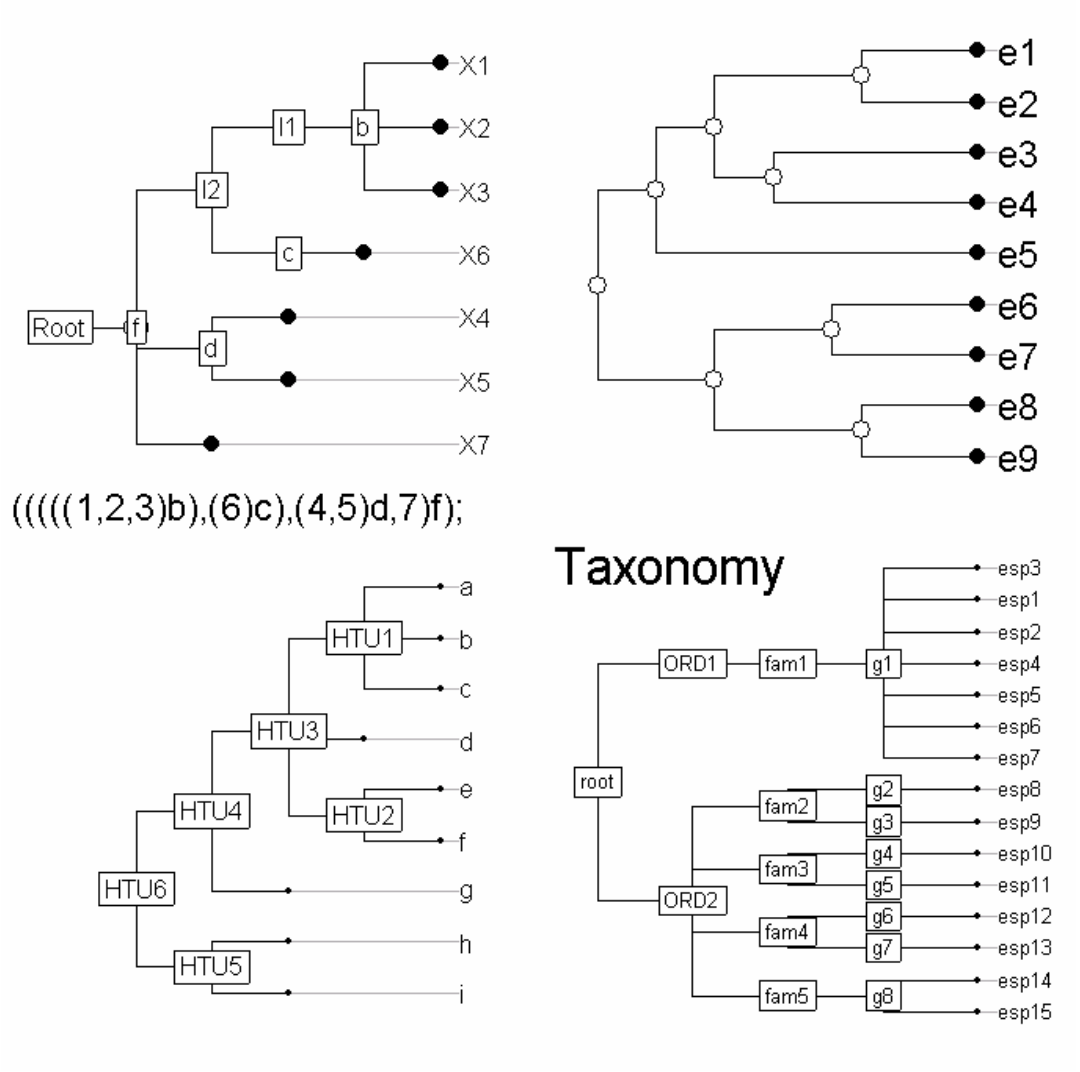
```
# plot.phylog
data(newick.eg)
par(mfrow = c(3,2))
for(i in 1:6) plot.phylog(newick2phylog(newick.eg[[i]],F),
       clea=2,clabel.l=3,cnod=2.5)
```



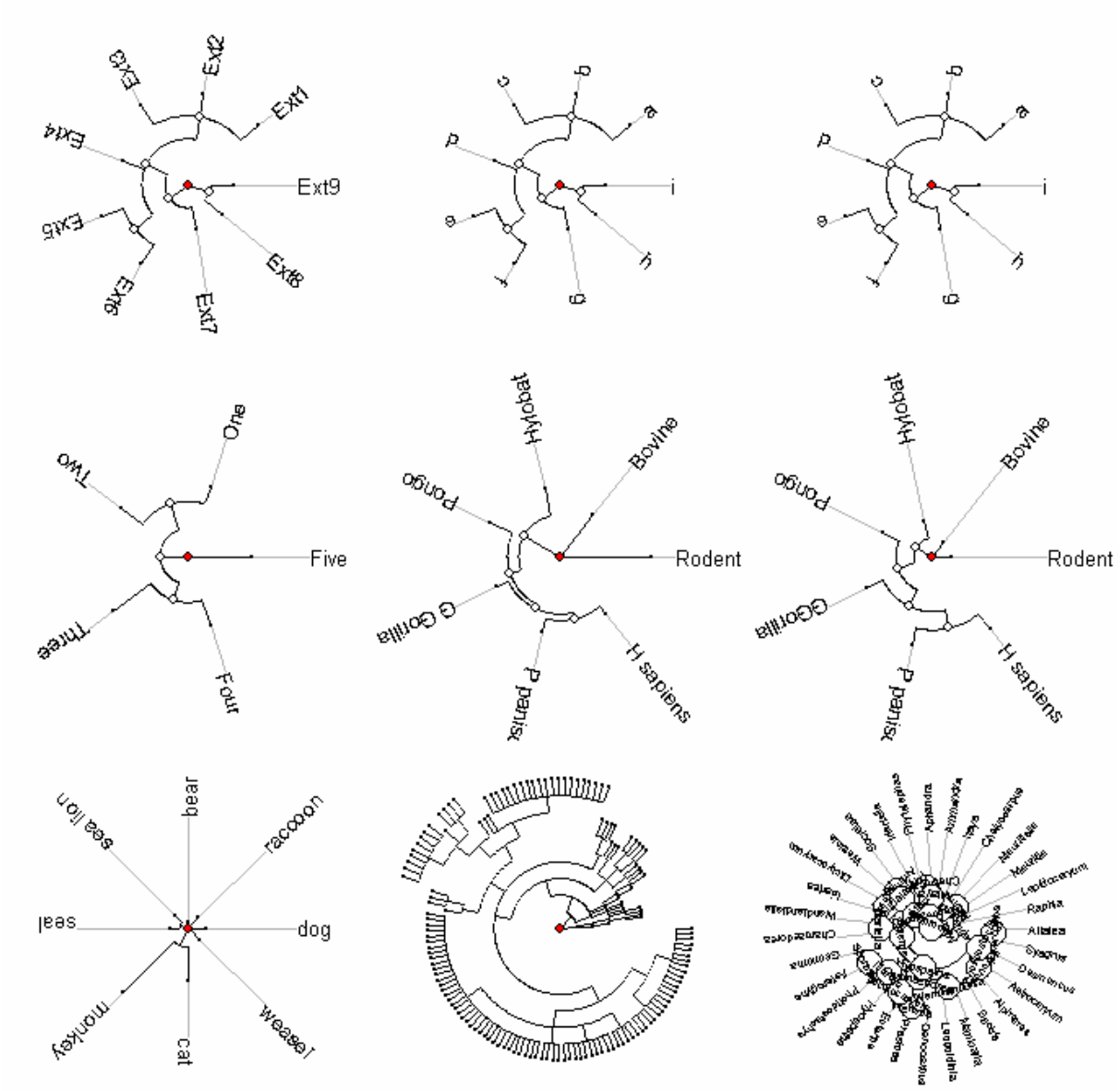

```
par(mfrow = c(1,2))
plot.phylog(newick2phylog(newick.eg[[11]],F),clea=1.5,
              clabel.l=1.5,clabel.nod=0.75,f=0.8)
plot.phylog(newick2phylog(newick.eg[[10]],F),clabel.l=0,clea=0,cn=0,f=1)
```



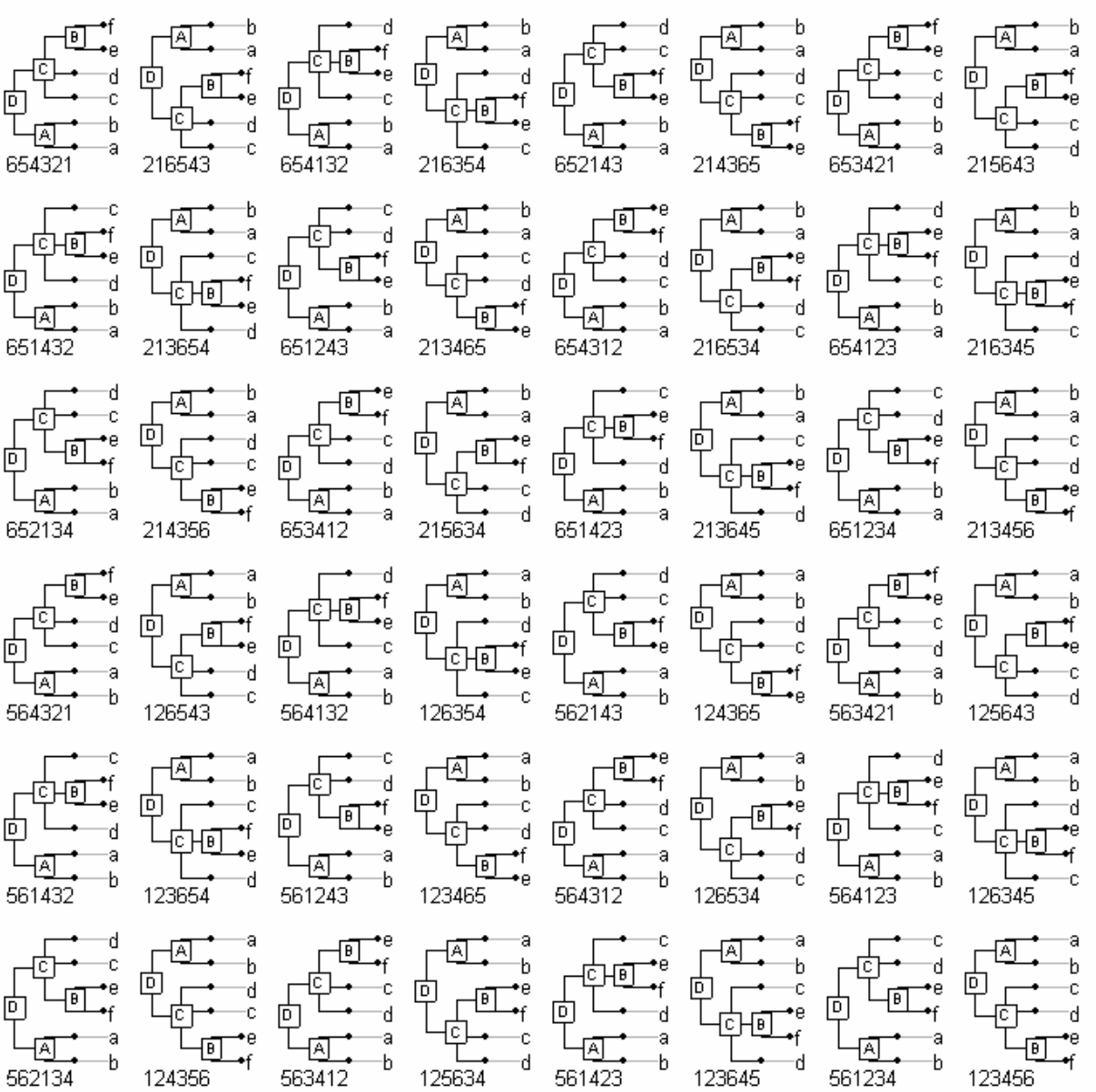
```
par(mfrow=c(2,2))
w7 = newick2phylog("(((((1,2,3)b),(6)c),(4,5)d,7)f);")
plot.phylog(w7,clabel.l=1.5,clabel.n=1.5,f=0.8,cle=2,cnod=3,
       sub="(((((1,2,3)b),(6)c),(4,5)d,7)f);",csub=2)

plot(newick2phylog("((((e1:4,e2:4)a:5,(e3:7,e4:7)b:2)c:2,e5:11)d:2,((e6:5,e7:5)e:4,
(e8:4,e9:4)f:5)g:4);"),f=0.8,cnod=2,cleav=2,clabel.l=2)

w = c("((((a:1,b:1,c:1)HTU1:1,d:1,(e:1,f:1)HTU2:1)HTU3:1,g:1",
")HTU4:1,(h:1,i:1)HTU5:1)HTU6:1;")
plot(newick2phylog(w),clea=1,cnod=1,clabel.l=1.5,clabel.n=1.5,f=0.8)

data(taxo.eg)
w = taxo2phylog(as.taxo(taxo.eg[[1]]))
plot(w,clabel.lea=1.25,clabel.n=1.25,sub="Taxonomy",csub=3,f=0.8,possub="topleft")
```




```
# radial.phylog
par(mfrow=c(3,3))
for (j in 1:6) radial.phylog(newick2phylog(newick.eg[[j]],F),clabel.l=2,cnodes=2)
radial.phylog(newick2phylog(newick.eg[[7]],F),clabel.l=2)
radial.phylog(newick2phylog(newick.eg[[8]],F),clabel.l=0,circle=1.8)
radial.phylog(newick2phylog(newick.eg[[9]],F),clabel.l=1,clabel.n =1,
       cle=0,cnode=1)
par(mfrow=c(1,1))
```




```
# enum.phylog
a <- "((a,b)A,(c,d,(e,f)B)C)D;"
wa <- newick2phylog(a)
wx <- enum.phylog(wa)
dim(wx)
# [1] 48  6
par(mfrow = c(6,8))
fun <- function(x) {
       w  <-NULL
       lapply(x,function(y) w<<-paste(w,as.character(y),sep=""))
       plot(wa,x,clabel.n=1.25,f=0.75,clabel.l=2,box=F,cle=1.5,sub=w,csub=2)
       invisible()}
apply(wx, 1, fun)
par(mfrow = c(1,1))
```



R code

```
"plot.phylog" <- function (x, y = NULL,
    f.phylog = 0.5, cleaves = 1, cnodes = 0,
    labels.leaves = names(x$leaves), clabel.leaves = 1,
    labels.nodes = names(x$nodes), clabel.nodes = 0,
    sub = "", csub = 1.25, possub = "bottomleft", draw.box = FALSE, ...)
 {
    if (!inherits(x, "phylog"))
        stop("Non convenient data")
    leaves.number <- length(x$leaves)
    leaves.names <- names(x$leaves)
    nodes.number <- length(x$nodes)
    nodes.names <- names(x$nodes)
    if (length(labels.leaves) != leaves.number) labels.leaves <- names(x$leaves)
    if (length(labels.nodes) != nodes.number) labels.nodes <- names(x$nodes)
    leaves.car <- gsub("[_]"," ",labels.leaves, ext = FALSE)
    nodes.car <- gsub("[_]"," ",labels.nodes, ext = FALSE)
    mar.old <- par("mar")
    on.exit(par(mar=mar.old))

    par(mar = c(0.1, 0.1, 0.1, 0.1))
```

```
    if (f.phylog < 0.05) f.phylog <- 0.05
    if (f.phylog > 0.95) f.phylog <- 0.95

    maxx <- max(x$droot)
    plot.default(0, 0, type = "n", xlab = "", ylab = "", xaxt = "n",
        yaxt = "n", xlim = c(-maxx*0.15, maxx/f.phylog), ylim = c(-0.05, 1), xaxs =
"i",
        yaxs = "i", frame.plot = FALSE)

    x.leaves <- x$droot[leaves.names]
    x.nodes <- x$droot[nodes.names]
    if (is.null(y)) y <- (leaves.number:1)/(leaves.number + 1)
    else y <- (leaves.number+1-y)/(leaves.number+1)
    names(y) <- leaves.names
    xcar <- maxx*1.05
    xx <- c(x.leaves, x.nodes)

    if (clabel.leaves > 0) {
        for (i in 1:leaves.number) {
            text(xcar, y[i], leaves.car[i], adj = 0, cex = par("cex") *
                clabel.leaves)
            segments(xcar, y[i], xx[i], y[i], col = grey(0.7))
        }
    }
    yleaves <- y[1:leaves.number]
    xleaves <- xx[1:leaves.number]
    if (cleaves > 0) {
        for (i in 1:leaves.number) {
            points(xx[i], y[i], pch = 21, bg=1, cex = par("cex") * cleaves)
        }
    }
    yn <- rep(0, nodes.number)
    names(yn) <- nodes.names
    y <- c(y, yn)
    for (i in 1:length(x$parts)) {
        w <- x$parts[[i]]
        but <- names(x$parts)[i]
        y[but] <- mean(y[w])
        b <- range(y[w])
        segments(xx[but], b[1], xx[but], b[2])
        x1 <- xx[w]
        y1 <- y[w]
        x2 <- rep(xx[but], length(w))
        segments(x1, y1, x2, y1)
    }
    if (cnodes > 0) {
        for (i in nodes.names) {
            points(xx[i], y[i], pch = 21, bg="white", cex = cnodes)
        }
    }
    if (clabel.nodes > 0) {
        scatterutil.eti(xx[names(x.nodes)], y[names(x.nodes)], labels.nodes,
            clabel.nodes)
    }
    x <- (x.leaves - par("usr")[1])/(par("usr")[2]-par("usr")[1])
    y <- y[leaves.names]
    xbase <- (xcar - par("usr")[1])/(par("usr")[2]-par("usr")[1])
    if (csub>0) scatterutil.sub(sub, csub=csub, possub=possub)
    if (draw.box) box()
    if (cleaves > 0) points(xleaves, yleaves, pch = 21, bg=1, cex = par("cex") *
cleaves)

    return(invisible(list(xy=data.frame(x=x, y=y), xbase= xbase, cleaves=cleaves)))
}

####   ####   ####   ####
####   ####   ####   ####
```

```
"radial.phylog" <- function (phylog, circle = 1,
    cleaves = 1, cnodes = 0,
    labels.leaves = names(phylog$leaves), clabel.leaves = 1,
    labels.nodes = names(phylog$nodes), clabel.nodes = 0,
    draw.box = FALSE)
{
    if (!inherits(phylog, "phylog"))
        stop("Non convenient data")
    leaves.number <- length(phylog$leaves)
    leaves.names <- names(phylog$leaves)
    nodes.number <- length(phylog$nodes)
    nodes.names <- names(phylog$nodes)
    if     (length(labels.leaves)     !=     leaves.number)     labels.leaves     <-
names(phylog$leaves)
    if (length(labels.nodes) != nodes.number) labels.nodes <- names(phylog$nodes)
    if (circle<0) stop("'circle': non convenient value")
    leaves.car <- gsub("[_]"," ",labels.leaves, ext = FALSE)
    nodes.car <- gsub("[_]"," ",labels.nodes, ext = FALSE)

    opar <- par(mar = par("mar"), srt = par("srt"))
    on.exit(par(opar))
    par(mar = c(0.1, 0.1, 0.1, 0.1))

    dis <- phylog$droot
    dis <- dis/max(dis)
    rayon <- circle
    dis <- dis * rayon
    dist.leaves <- dis[leaves.names]
    dist.nodes <- dis[nodes.names]
    plot.default(0, 0, type = "n", asp = 1, xlab = "", ylab = "",
        xaxt = "n", yaxt = "n", xlim = c(-2, 2), ylim = c(-2,
            2), xaxs = "i", yaxs = "i", frame.plot = FALSE)
    d.rayon <- rayon/(nodes.number - 1)
    alpha <- 2 * pi * (1:leaves.number)/leaves.number
    names(alpha) <- leaves.names
    x <- dist.leaves * cos(alpha)
    y <- dist.leaves * sin(alpha)
    xcar <- (rayon + d.rayon) * cos(alpha)
    ycar <- (rayon + d.rayon) * sin(alpha)
    if (clabel.leaves>0) {
        for (i in 1:leaves.number) {
            segments(xcar[i], ycar[i], x[i], y[i], col = grey(0.7))
        }
        for (i in 1:leaves.number) {
            par(srt = alpha[i] * 360/2/pi)
            text(xcar[i], ycar[i], leaves.car[i], adj = 0, cex = par("cex") *
                clabel.leaves)
            segments(xcar[i], ycar[i], x[i], y[i], col = grey(0.7))
        }
    }
    if (cleaves > 0) {
        for (i in 1:leaves.number) points(x[i], y[i], pch = 21, bg="black", cex =
par("cex") *
            cleaves)
    }
    ang <- rep(0, length(dist.nodes))
    names(ang) <- names(dist.nodes)
    ang <- c(alpha, ang)
    for (i in 1:length(phylog$parts)) {
        w <- phylog$parts[[i]]
        but <- names(phylog$parts)[i]
        ang[but] <- mean(ang[w])
        b <- range(ang[w])
        a.seq <- c(seq(b[1], b[2], by = pi/180), b[2])
        lines(dis[but] * cos(a.seq), dis[but] * sin(a.seq))
        x1 <- dis[w] * cos(ang[w])
        y1 <- dis[w] * sin(ang[w])
        x2 <- dis[but] * cos(ang[w])
```

```
        y2 <- dis[but] * sin(ang[w])
        segments(x1, y1, x2, y2)
    }
    if (cnodes > 0) {
        for (i in 1:length(phylog$parts)) {
            w <- phylog$parts[[i]]
            but <- names(phylog$parts)[i]
            ang[but] <- mean(ang[w])
             points(dis[but] * cos(ang[but]), dis[but] * sin(ang[but]),
                pch = 21, bg="white", cex = par("cex") * cnodes)
        }
    }
    points(0, 0, pch = 21, cex = par("cex") * 2, bg = "red")
    if (clabel.nodes > 0) {
        delta <- strwidth(as.character(length(dist.nodes)), cex = par("cex") *
            clabel.nodes)
        for (j in 1:length(dist.nodes)) {
            i <- names(dist.nodes)[j]
            par(srt = (ang[i] * 360/2/pi + 90))
            x1 <- dis[i] * cos(ang[i])
            y1 <- dis[i] * sin(ang[i])
            symbols(x1, y1, delta, bg = "white", add = TRUE, inch = FALSE)
            text(x1, y1, nodes.car[j], adj = 0.5, cex = par("cex") *
                clabel.nodes)
        }
    }
    if (draw.box) box()
    return(invisible())
}

####   ####   ####   ####
####   ####   ####   ####

enum.phylog<-function (phylog, no.over=1000) {

    # Pour chaque phylogénie phylog, il existe un grand nombre de représentations
    # toutes équivalentes associées à la même topologie
    # Il y en a exactement 2^k pour une phylogénie résolue
    # (que des dichotomies), ou k représente le nombre de noeuds
    # Cette fonction énumère tous les possibles
    if (!inherits(phylog, "phylog")) stop("Object 'phylog' expected")
    leaves.number<- length(phylog$leaves)
    leaves.names<- names(phylog$leaves)
    # les descendants sont pris par la racine
    parts <- rev(phylog$parts)
    nodes.number<- length(parts)
    nodes.names<- (names(parts))
    nodes.dim <- unlist(lapply(parts,length))
    perms.number <- prod(gamma(nodes.dim+1))
    if (perms.number>no.over) {
        cat("Permutation number =",perms.number,"( no.over =", no.over,")\n")
        return(invisible())
    }

    "perm" <- function(cha=as.character(1:n),a=matrix(1,1,1)) {
        n0 = ncol(a)
        n = length(cha)
        if (n0 == n) {
            a <- apply(a,c(1,2),function(x) cha[x])
            return(a)
        }
        fun1 <- function(x) {
                xplus = length(x)+1
                fun2 <- function (j) {
                        if (j==1) w= c(xplus,x)
                        else if (j==xplus) w = c(x,xplus)
                        else w = c(x[1:j-1],xplus,x[j:length(x)])
                        return(w)
```

```
                }
                w = sapply(1:(length(x)+1) , fun2)
        }
        a = matrix(unlist(apply(a,1,fun1)),ncol=n0+1,byr=TRUE)
        Recall(cha,a)
    }

    res <- matrix (1,1,1)

    lw <- lapply(parts,perm)
    names(lw) <- nodes.names
    res <- lw[[1]]

    lw[[1]]<- NULL

    "permtot" <- function (matcar) {
        n1 <- nrow(res) ; n2 <- nrow(matcar)
        p1 <- ncol(res) ; p2 <- ncol(matcar)
        f1 <- function(x) unlist(apply(res,1,function(y) c(y,x)))
        res <<- matrix(unlist(apply(matcar,1,f1)),n1*n2, p1+p2,byr=TRUE)
    }

    lapply(lw, permtot)

     ################################################
    permut<-function(init, node){
        # init est un vecteur des noms des feuilles compatibles avec la phylogénie
        # on opère une permutation circulaire des blocs de feuilles associés
        # aux descendants immédiats du noeud node

        num<- grep(node, nodes.names)[1]
        w0 <- enfants(node,res)
        if (min(w0)>1) a <- 1:(min(w0)-1) else a <- NULL
        if (max(w0)<leaves.number) b <- (max(w0)+1):leaves.number else b <- NULL
        agauche<-
which(unlist(phylog$parts[node])%in%unlist(phylog$paths[init[w0[1]]]))
        w1 <- unlist(lapply(phylog$parts[[num]][agauche],enfants, init=res))
        w2 <- unlist(lapply(phylog$parts[[num]][-agauche],enfants, init=res))
        return(init[c(a,w2,w1,b)])
    }
    ################################################
    fac <- factor(rep(1:nodes.number,nodes.dim))
    renum <- function (cha) {
        cha <- split(cha, fac)
        names(cha) <- nodes.names
        w <- cha[[1]]
        for (j in nodes.names[-1]) {
              k <- which(w==j)
              wcha <- cha[[j]]
              if (k==1) w <- c(wcha,w[-k])
              else if (k == length(w)) w <- c(w[-k],wcha)
              else w <- c(w[1:(k-1)],wcha,w[(k+1):length(w)])
        }
        res <- 1:leaves.number
        names(res)=w
        return(res[leaves.names])
    }
    return(t(apply(res,1,renum)))


}
```

21. Typologie de formes bilinéaires symétriques

alias

statis.kfbs

description

'statis.kfbs' définit une typologie de formes bilinéaires symétriques. Elle permet d'étudier la redondance implicite des points de vue dans une famille de formes bilinéaires symétriques.

usage

statis.kfbs(kfbs)

arguments

- *kfbs* : est un tableau de la classe 'kfbs'

details

Pour comparer deux métriques, on prendra, comme pour l'approche STATIS, le cosinus

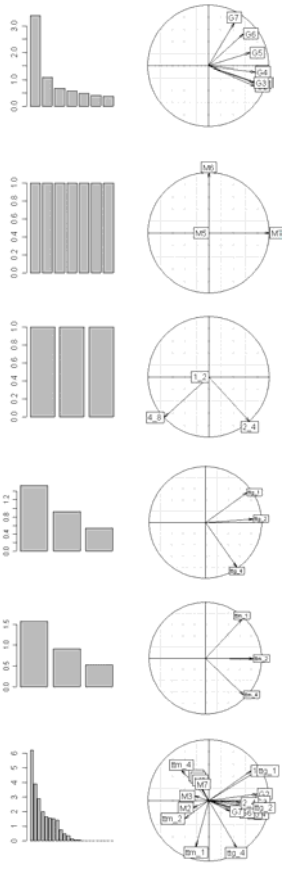
$$RV_{ij} = RV_{ji} = \cos\left(\mathbf{A}_i, \mathbf{A}_j\right) = \frac{trace\left(\mathbf{A}_i\mathbf{A}_j\right)}{\sqrt{trace\left(\mathbf{A}_i^2\right)trace\left(\mathbf{A}_j^2\right)}}$$

valeurs

'statis.kfbs' retourne la matrice de terme générale $\left(RV_{ij}\right)_{1\le i\le k}^{1\le j\le k}$, les valeurs propres et les coordonnées des métriques.

exemples

```
ng <- neig(n.line = 8)     # graphe linéaire
knb <- neig2knb(ng)        # puissances du graphes

geary.kfbs <- knb2kfbs(knb, method = "Geary") # formes de Geary/Lebart
moran.kfbs <- knb2kfbs(knb, method = "Moran") # formes de Moran/Smouse
noy.kfbs <- msbs.kfbs(n = 8, tbloc = c(1,2,4,8))     # formes de Noy-Meir/Anderson
hill.geary.kfbs <- ttlv.kfbs(n=8,tbloc=c(1,2,4),method = "Geary") # formes de Hill
hill.moran.kfbs <- ttlv.kfbs(n=8,tbloc=c(1,2,4),method="Moran") # formes de Hill


geary.statis <- statis.kfbs(geary.kfbs)
par(mfrow = c(1,2))
barplot(geary.statis$ev)
s.corcircle(geary.statis$co)

moran.statis <- statis.kfbs(moran.kfbs)
par(mfrow = c(1,2))
barplot(moran.statis$ev)
s.corcircle(moran.statis$co)

noy.statis <- statis.kfbs(noy.kfbs)
par(mfrow = c(1,2))
barplot(noy.statis$ev)
s.corcircle(noy.statis$co)

hill.geary.statis <- statis.kfbs(hill.geary.kfbs)
hill.moran.statis <- statis.kfbs(hill.moran.kfbs)

par(mfrow = c(2,2))
barplot(hill.geary.statis$ev)
s.corcircle(hill.geary.statis$co)
barplot(hill.moran.statis$ev)
s.corcircle(hill.moran.statis$co)

all.kfbs <- cbind.kfbs(geary.kfbs, moran.kfbs,
 noy.kfbs, hill.geary.kfbs, hill.moran.kfbs)
all.statis <- statis.kfbs(all.kfbs)
par(mfrow = c(1,2))
barplot(all.statis$ev)
s.corcircle(all.statis$co, 1 ,2)
```



R code

```
"statis.kfbs" <- function(kfbs){
    if (class(kfbs)!="kfbs") stop ("Non convenient data")

n <- attr(kfbs, "npoints")
p <- attr(kfbs, "nforms")
trace1 <- attr(kfbs, "trace1")
trace1 <- attr(kfbs, "trace1")
tab <- diag(1, n)
tab <- as.matrix(tab)
sxx <- tab%*%t(tab)
sxx <- sxx/sum(diag(sxx))
psX <- matrix(0, p, p)

for (i in 1:p) {
    a <- matrix(0, n, n)
    a[row(a) >= col(a)] <- kfbs[, i]
    a <- a + t(a)
    a <- a-diag(diag(a)/2)
    ax <- sxx%*%a
    for (j in i:p) {
        b <- matrix(0, n, n)
        b[row(b) >= col(b)] <- kfbs[, j]
        b <- b + t(b)
        b <- b-diag(diag(b)/2)
        b <- sxx%*%b
        psX[i, j] <- sum(diag(ax%*%b))
        psX[j, i] <- psX[i, j]
    }
}

psX <- psX/sqrt(matrix(diag(psX), p, p, byr = TRUE)* matrix(diag(psX), p, p))
eig <- eigen(psX)
row.names(psX) <- attr(kfbs, "labels")
names(psX) <- attr(kfbs, "labels")
w <- sqrt(eig$values^2)
co <- eig$vectors*matrix(sqrt(w), p, p, byr = TRUE)
if (max(co[, 1]<0)) co[, 1] <- -co[, 1]
row.names(co) <- attr(kfbs, "labels")
res <- list(RV = psX, ev = w, co = co)
return(res)
}
```

22. Représentation graphique d'un trait biologique dans un arbre phylogénétique

alias

symbols.phylog

description

'symbols.phylog' représente la phylogénie verticalement et la valeur de la variable par un symbole (carré ou cercle) de taille proportionnelle à la valeur (racine de la surface proportionnelle à la valeur absolue de la valeur). N'utiliser cette fonction que pour des variables centrées (symbole blanc en dessous de la moyenne, symbole noir au-dessus de la moyenne).

usage

```
symbols.phylog(phylog, y = NULL, l.circles, l.squares, n.circles, n.squares, csize = 1, clegend = 1, sub = "", csub = 1, possub = "topleft")
```

arguments

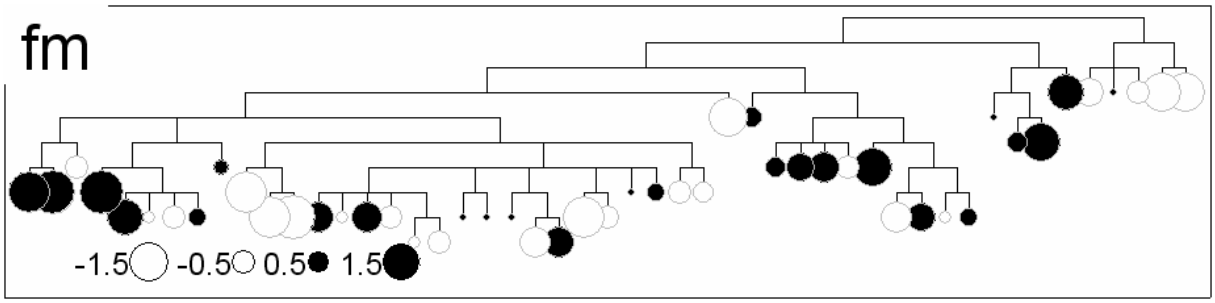
- *phylog* : est un objet de la classe 'phylog'
- *y* : est un vecteur donnant la position des feuilles pour une représentation donnée de la phylogénie (voir, la fonction enum.phylog)
- *l.circles* : est un vecteur donnant les valeurs portées par les feuilles représentées par des cercles
- *l.squares* : est un vecteur donnant les valeurs portées par les feuilles représentées par des carrés
- *n.squares* : est un vecteur donnant les valeurs portées par les noeuds représentées par des cercles
- *l.squares* : est un vecteur donnant les valeurs portées par les noeuds représentées par des carrés
- *csize* : est un entier donnant la taille des symboles. Si 0, seule la phylogénie est tracée.
- *clegend* : est un entier donnant la taille des caractères de la légende. Si 0, il n'y a pas de légende.
- *sub* : est une chaîne de caractères pour le titre
- *csub* : est un entier donnant la taille des caractères du titre
- *possub* : est une chaîne de caractères donnant la position du titre

voir également

dotchart.phylog
table.phylog

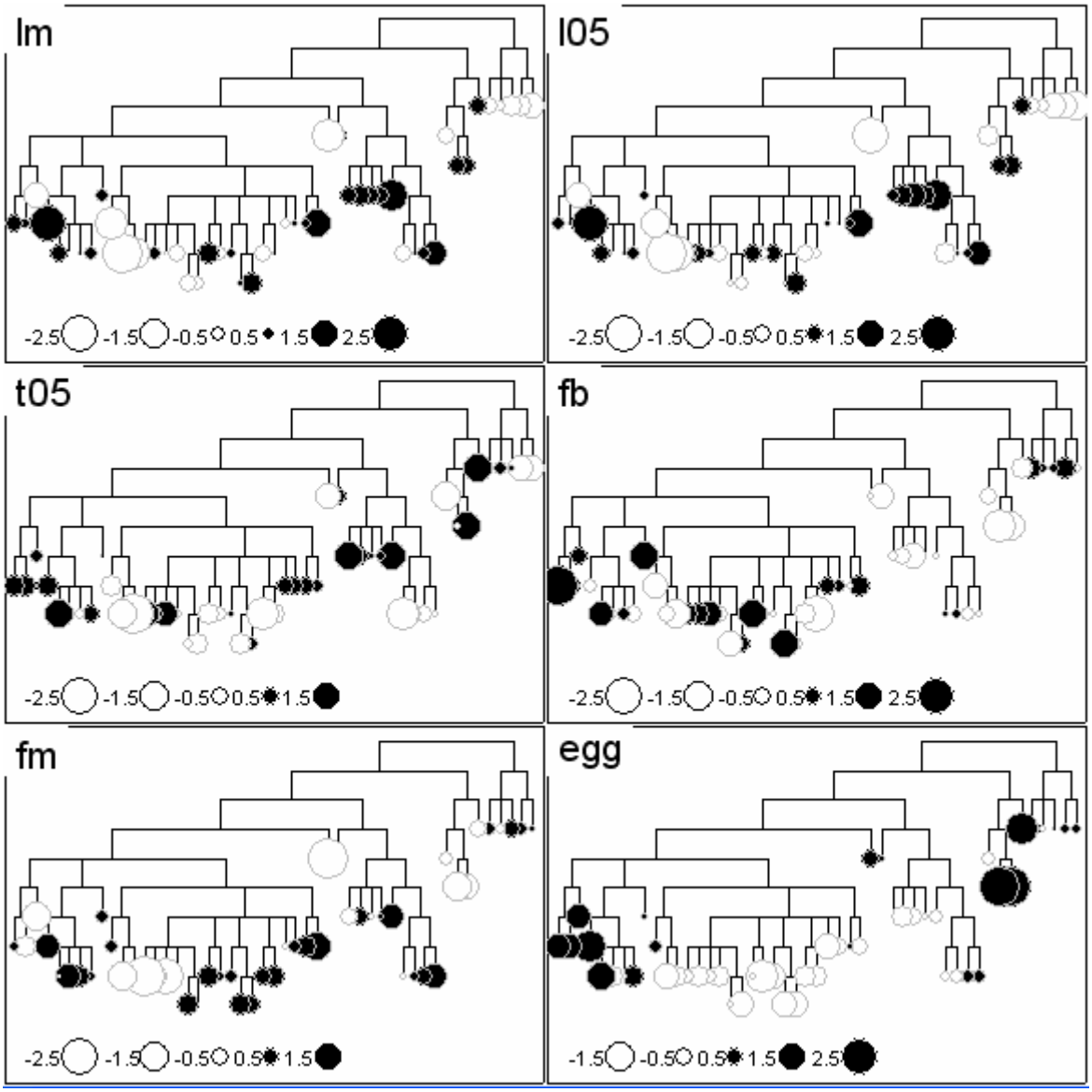
exemples

```
mjrochet.phy <- newick2phylog(mjrochet$tre)
tab0 <- data.frame(scalewt(log(mjrochet$tab)))
symbols.phylog(phylog = mjrochet.phy,
 l.circles = tab0[,1], csi = 0.75, cleg = 1.5,
 sub = names(tab0)[j], csub = 3)
```




```
par(mfrow = c(3,2))
for (j in 2:7) {
       w <- tab0[,j]
       symbols.phylog(phylog = mjrochet.phy,
 l.circles = w, csi = 1.5, cleg = 1.5,
 sub = names(tab0)[j],csub=3)
```

```
}
```



R code

```
"symbols.phylog" <- function (phylog, y = NULL, l.circles, l.squares, n.circles,
n.squares, csize = 1, clegend = 1,
 sub = "", csub = 1, possub = "topleft")
{
    if (!inherits(phylog, "phylog"))
        stop("Non convenient data")
    l.count <- 0
    l.data <- NULL
    if (!missing(l.circles)) {
        l.count <- l.count + 1
        l.data <- l.circles
        l.type <- 2
    }
    if (!missing(l.squares)) {
        l.count <- l.count + 1
        l.data <- l.squares
        l.type <- 1
    }

    n.count <- 0
    n.data <- NULL
```

```
    if (!missing(n.circles)) {
        n.count <- n.count + 1
        n.data <- n.circles
        n.type <- 2
    }
    if (!missing(n.squares)) {
        n.count <- n.count + 1
        n.data <- n.squares
        n.type <- 1
    }

    if (l.count > 1 || n.count > 1)
        stop("no more than one symbol type must be specified for leaves or nodes")
    if (csize <= 0) {
        l.data <- NULL
        n.data <- NULL
    }
    if (!is.null(l.data)) {
        if (is.null(names(l.data)))
            names(l.data) <- names(phylog$leaves)
        if (length(l.data) != length(phylog$leaves)) l.data <- NULL
        if (!is.null(l.data)) {
            w1 <- sort(names(l.data))
            w2 <- sort(names(phylog$leaves))
            if (!all(w1 == w2)) {
                print(w1)
                print(w2)
                warning("names(data) non convenient for 'phylog' : we use the names
of the leaves in 'phylog'")
                names(l.data) <- names(phylog$leaves)
            }
            l.data <- l.data[names(phylog$leaves)]
        }
    }

    if (!is.null(n.data)) {
        if (is.null(names(n.data)))
            names(n.data) <- names(phylog$nodes)
        if (length(n.data) != length(phylog$nodes)) n.data <- NULL
        if (!is.null(n.data)) {
            w1 <- sort(names(n.data))
            w2 <- sort(names(phylog$nodes))
            if (!all(w1 == w2)) {
                print(w1)
                print(w2)
                warning("names(data) non convenient for 'phylog' : we use the names
of the nodes in 'phylog'")
                names(n.data) <- names(phylog$nodes)
            }
            n.data <- n.data[names(phylog$nodes)]
        }
    }

    opar <- par(mar = par("mar"))
    on.exit(par(opar))
    par(mar = c(0.1, 0.1, 0.1, 0.1))
    plot.default(0, 0, type = "n", xlab = "", ylab = "", xaxt = "n",
        yaxt = "n", ylim = c(-0.2, 1.05), xlim = c(0, 1), xaxs = "i",
        yaxs = "i", frame.plot = TRUE)

    symbol.max <- csize/20
    if (symbol.max > 0.5)
        symbol.max <- 0.5
    dis <- phylog$droot
    dis <- 1 - ((1 - symbol.max) * dis/max(dis))
    xinit <- dis[names(phylog$leaves)]
    dn <- dis[names(phylog$nodes)]
    n <- length(xinit)
```

```
if (is.null(y))
    yinit <- (n:1)/(n + 1)
else yinit <- (n + 1 - y)/(n + 1)
names(yinit) <- names(phylog$leaves)
x <- dis
yn <- rep(0, length(dn))
names(yn) <- names(dn)
y <- c(yinit, yn)
yinit.n <- yinit
xinit.n <- (min(xinit)-par("usr")[3]/2)/4
xinit.n <- rep(min(xinit), n) - xinit.n

legender <- function(br0, sq0, sig0, clegend, type) {
    br0 <- round(br0, dig = 6)
    cha <- as.character(br0[1])
    for (i in (2:(length(br0)))) cha <- paste(cha, br0[i],
        sep = " ")
    cex0 <- par("cex") * clegend
    yh <- max(c(strheight(cha, cex = cex0), sq0))
    h <- strheight(cha, cex = cex0)
    y0 <- par("usr")[3] + yh/2 + h
    ltot <- strwidth(cha, cex = cex0) + sum(sq0) + h
    x0 <- par("usr")[1] + h/2
    for (i in (1:(length(sq0)))) {
        cha <- br0[i]
        cha <- paste(" ", cha, sep = "")
        xh <- strwidth(cha, cex = cex0)
        text(x0 + xh/2, y0, cha, cex = cex0)
        z0 <- sq0[i]
        x0 <- x0 + xh + z0/2
        if (sig0[i] >= 0) {
            if (type == 1)
              symbols(x0, y0, squares = z0, bg = "black",
                fg = "white", add = TRUE, inch = FALSE)
            else if (type == 2)
              symbols(x0, y0, circles = z0/2, bg = "black",
                fg = "white", add = TRUE, inch = FALSE)
        }
        else {
            if (type == 1)
              symbols(x0, y0, squares = z0, bg = "white",
                fg = "black", add = TRUE, inch = FALSE)
            else if (type == 2)
              symbols(x0, y0, circles = z0/2, bg = "white",
                fg = "black", add = TRUE, inch = FALSE)
        }
        x0 <- x0 + z0/2
    }
    invisible()
}

    legender <- function(br0, sq0, sig0, clegend, type) {
    br0 <- round(br0, dig = 6)
    cha <- as.character(br0[1])
    for (i in (2:(length(br0)))) cha <- paste(cha, br0[i],
        sep = " ")
    cex0 <- par("cex") * clegend
    yh <- max(c(strheight(cha, cex = cex0), sq0))
    h <- strheight(cha, cex = cex0)
    y0 <- par("usr")[3] + yh/2 + h
    ltot <- strwidth(cha, cex = cex0) + sum(sq0) + h
    x0 <- par("usr")[1] + h/2
    for (i in (1:(length(sq0)))) {
        cha <- br0[i]
        cha <- paste(" ", cha, sep = "")
        xh <- strwidth(cha, cex = cex0)
        text(x0 + xh/2, y0, cha, cex = cex0)
        z0 <- sq0[i]
```

```
            x0 <- x0 + xh + z0/2
            if (sig0[i] >= 0) {
                if (type == 1)
                  symbols(x0, y0, squares = z0, bg = "black",
                    fg = "white", add = TRUE, inch = FALSE)
                else if (type == 2)
                  symbols(x0, y0, circles = z0/2, bg = "black",
                    fg = "white", add = TRUE, inch = FALSE)
            }
            else {
                if (type == 1)
                  symbols(x0, y0, squares = z0, bg = "white",
                    fg = "black", add = TRUE, inch = FALSE)
                else if (type == 2)
                  symbols(x0, y0, circles = z0/2, bg = "white",
                    fg = "black", add = TRUE, inch = FALSE)
            }
            x0 <- x0 + z0/2
        }
        invisible()
    }

    for (i in 1:length(phylog$parts)) {
        w <- phylog$parts[[i]]
        but <- names(phylog$parts)[i]
        y[but] <- mean(y[w])
        b <- range(y[w])
        segments(b[1], x[but], b[2], x[but], lwd = 1)
        x1 <- x[w]
        y1 <- y[w]
        x2 <- rep(x[but], length(w))
        segments(y1, x1, y1, x2, lwd = 1)
    }

    if (!is.null(l.data) && !is.null(n.data)) {
        for(i in 1:n)
            segments(yinit, xinit, yinit.n, xinit.n, col = grey(0.75), lwd = 0.5)
    }

    xnodes <- x[names(phylog$nodes)]
    ynodes <- y[names(phylog$nodes)]
    l <- length(names(phylog$nodes))

if (!is.null(l.data)) {
    if (!is.factor(l.data)){
        if(!is.null(n.data)) {
        yinit <- yinit.n
        xinit <- xinit.n
        }
        sq <- sqrt(abs(l.data))
        w1 <- max(sq)
        sq <- symbol.max * sq/w1
        if (l.type == 1) {
            for (i in 1:n) {
                if (sign(l.data[i]) >= 0) {
                  symbols(yinit[i], xinit[i], squares = sq[i],
                    bg = "black", fg = "grey", add = TRUE, inch = FALSE)
                }
                else {
                  symbols(yinit[i], xinit[i], squares = sq[i],
                    bg = "white", fg = "grey", add = TRUE, inch = FALSE)
                }
            }
        }
        else if (l.type == 2) {
            for (i in 1:n) {
                if (sign(l.data[i]) >= 0) {
                  symbols(yinit[i], xinit[i], circles = sq[i]/2,
```

```
                    bg = "black", fg = "grey", add = TRUE, inch = FALSE)
                }
                else {
                  symbols(yinit[i], xinit[i], circles = sq[i]/2,
                    bg = "white", fg = "grey", add = TRUE, inch = FALSE)
                }
            }
        }
        if (clegend > 0) {
            br0 <- pretty(l.data, 4)
            l0 <- length(br0)
            br0 <- (br0[1:(l0 - 1)] + br0[2:l0])/2
            sq0 <- sqrt(abs(br0))
            sq0 <- symbol.max * sq0/w1
            sig0 <- sign(br0)
            legender(br0, sq0, sig0, clegend = clegend, type = l.type)
        }
    }

    else{
    if(!is.null(n.data)) {
        yinit <- yinit.n
        xinit <- xinit.n
        }
    lev <- levels(l.data)
    nlev <- nlevels(l.data)
    if (l.type == 1) {
            for (i in 1:n)
                symbols(yinit[i], xinit[i], squares = csize / 20,
                    bg = (1:nlev)[l.data[i] == lev], fg = "grey", add = TRUE, inch
= FALSE)
                }

        else if (l.type == 2) {
            for (i in 1:n)
                symbols(yinit[i], xinit[i], circles = csize / 20,
                    bg = (1:nlev)[l.data[i] == lev], fg = "grey", add = TRUE, inch
= FALSE)
                }
    if (clegend > 0) {
            cex0 <- par("cex") * clegend
            h <- strheight(lev, cex = cex0)
            y0 <- par("usr")[3] + 3*h/2
            x0 <- par("usr")[1] + h/2
        for (i in 1:nlev) {
            cha <- lev[i]
            cha <- paste(" ", cha, sep = "")
            xh <- strwidth(cha, cex = cex0)
            text(x0 , y0, cha, cex = cex0, col = i)
            #symbols(x0 + xh, y0, circles = csize/30, bg = i, fg = "grey", add =
TRUE, inch = FALSE)
            x0 <- x0 + xh + csize/30 + h/2
            }
        }
    }
}
    if (!is.null(n.data)) {
        sq <- sqrt(abs(n.data))
        w1 <- max(sq)
        sq <- symbol.max * sq/w1
        if (n.type == 1) {
            for (i in 1:l) {
                if (sign(n.data[i]) >= 0) {
                  symbols(ynodes[i], xnodes[i], squares = sq[i],
                    bg = "black", fg = "grey", add = TRUE, inch = FALSE)
                }
                else {
                  symbols(ynodes[i], xnodes[i], squares = sq[i],
```

```
                    bg = "white", fg = "grey", add = TRUE, inch = FALSE)
                }
            }
        }
        else if (n.type == 2) {
            for (i in 1:l) {
                if (sign(n.data[i]) >= 0) {
                  symbols(ynodes[i], xnodes[i], circles = sq[i]/2,
                    bg = "black", fg = "grey", add = TRUE, inch = FALSE)
                }
                else {
                  symbols(ynodes[i], xnodes[i], circles = sq[i]/2,
                    bg = "white", fg = "grey", add = TRUE, inch = FALSE)
                }
            }
        }
        if ((clegend > 0) && (is.null(l.data))) {
            br0 <- pretty(n.data, 4)
            l0 <- length(br0)
            br0 <- (br0[1:(l0 - 1)] + br0[2:l0])/2
            sq0 <- sqrt(abs(br0))
            sq0 <- symbol.max * sq0/w1
            sig0 <- sign(br0)
            legender(br0, sq0, sig0, clegend = clegend, type = n.type)
        }
    }

    if (csub > 0)
        scatterutil.sub(sub, csub, possub)
}
```

23. Représentation graphique de plusieurs traits dans un arbre phylogénétique

alias

table.phylog

description

'table.phylog' représente la phylogénie verticalement et les valeurs de plusieurs traits par des carrés de taille proportionnelle à la valeur (racine de la surface proportionnelle à la valeur absolue de la valeur). On utilise cette fonction préférentiellement sur des variables centrées (symbole blanc en dessous de la moyenne, symbole noir au-dessus de la moyenne).

usage

```
table.phylog(df, phylog, x = 1:ncol(df), y = NULL, f.phylog = 0.5,
    labels.row = gsub("[_]", " ", row.names(df)), clabel.row = 1,
    labels.col = names(df), clabel.col = 1, labels.nod = names(phylog$nodes),
    clabel.nod = 0, cleaves = 1, cnodes = 1, csize = 1, grid = TRUE,
    clegend = 0.75)
```

arguments

- df : est un data frame ne comportant que des variables numériques
- phylog : est un objet de la classe 'phylog'
- y : est un vecteur donnant la position des feuilles pour une représentation donnée de la phylogénie (voir, la fonction enum.phylog)
- x : est un vecteur de coordonnées pour positionner les colonnes du tableau
- f.phylog : est un entier donnant la taille (en pourcentage) réservée à la représentation de la phylogénie
- labels.row : est un vecteur des étiquettes des lignes (feuille de la phylogénie), par défaut ce sont les noms des feuilles avec ' ' à la place de '_'
- clabel.row : est un entier donnant la taille des caractères des lignes
- labels.col : est un vecteur des étiquettes des colonnes (variables du data frame)
- clabel.col : est un entier donnant la taille des caractères des colonnes
- labels.nod : est un vecteur des étiquettes des noeds de la phylogénie
- clabel.nod : est un entier donnant la taille des caractères des nœuds
- csize : est un entier donnant la taille des carrés pour les valeurs dans le tableau
- grid : est une variable binaire.Si TRUE la grille du tableau est tracée
- clegend : est un entier donnant la taille des caractères de la légende. Si 0, il n'y a pas de légende pour la taille des carrés.

détails

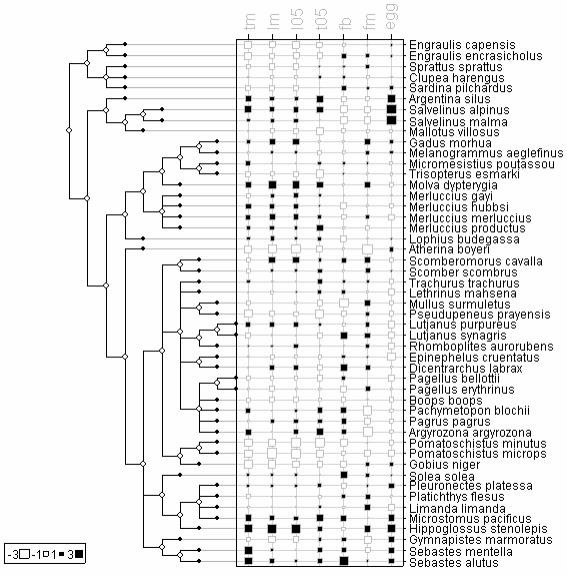
Le data frame doit avoir des row.names identiques à ceux des feuilles de la phylogénie.

voir également

symbols.phylog
table.pylog

exemples

```
mjrochet.phy <- newick2phylog(
 mjrochet$tre)
tab0 <- data.frame(
 scalewt(log(mjrochet$tab)))
table.phylog(tab0,
 mjrochet.phy,
 clabel.row = 0.75)
```



R code

```
"table.phylog" <- function (df, phylog, x = 1:ncol(df), y = NULL, f.phylog = 0.5,
labels.row = gsub("[_]", " ", row.names(df)), clabel.row = 1, labels.col =
names(df), clabel.col = 1, labels.nod = names(phylog$nodes), clabel.nod = 0,
cleaves = 1, cnodes = 1, csize = 1, grid = TRUE, clegend = 0.75)
{
    if (!inherits(df, "data.frame"))
        stop("data.frame expected for 'df'")
    if (!inherits(phylog, "phylog"))
        stop("class 'phylog' expected for 'phylog'")

    df.list <- as.list(df)
    nvar <- length(df.list)
    fac.num <- (1:nvar)[unlist(lapply(df.list, is.factor))]
    fac.length <- unlist(lapply(df.list[fac.num], nlevels))
    col.col <- rep(0, nvar - length(fac.num))
    fac.fg <- rep(col.col, rep(nrow(df), length(col.col)))
    if(length(fac.num) != 0){
    u <- 1:length(fac.num)
    v <- rep(u, fac.length)
    w <- nvar-length(fac.num)
    col.col <- rep(0, w + length(v))
    pos <- 0
    for(i in 1:length(fac.num)){
        col.col[(fac.num[i] + pos):(fac.num[i] + pos + fac.length[i] -1)] <- rep(i,
fac.length[i])
        pos <- pos + fac.length[i] - 1
        }
    fac.fg <- rep(col.col, rep(nrow(df), length(col.col)))
    df.list       <-       lapply(df.list,       function(x)       if(is.factor(x))
acm.disjonctif(as.data.frame(x)) else x)
    }
    res <- as.data.frame.list(df.list)
    row.names(res) <- row.names(df)
    df <- res
    leave.names <- names(phylog$leaves)
    node.names <- names(phylog$nodes)
    n.leave <- length(leave.names)
    n.node <- length(node.names)
    if (f.phylog > 0.8)
        f.phylog <- 0.8
    if (f.phylog < 0.2)
        f.phylog <- 0.2
    opar <- par(mai = par("mai"), srt = par("srt"))
    on.exit(par(opar))
    w1 <- sort(row.names(df))
    w2 <- sort(names(phylog$leaves))
    if (!all(w1 == w2)) {
        print.noquote("names from 'df'")
        print(w1)
        print.noquote("names from 'phylog'")
        print(w2)
        stop("non convenient matching information")
    }
    df <- df[names(phylog$leaves), ]
    frame()
    labels.row <- paste(" ", labels.row, " ", sep = "")
    labels.col <- paste(" ", labels.col, " ", sep = "")
    cexrow <- par("cex") * clabel.row
    strx <- 0.1
    if (cexrow > 0) {
        strx <- max(strwidth(labels.row, unit = "inches", cex = cexrow)) +
            0.1
    }
    cexcol <- par("cex") * clabel.col
    stry <- 0.1
    if (cexcol > 0) {
        stry <- max(strwidth(labels.col, unit = "inches", cex = cexcol)) +
```

```
        0.1
}
par(mai = c(0.1, 0.1, stry, strx))
nc <- ncol(df)
x <- 1/2/nc + (0:(nc - 1))/nc
x <- (1 - f.phylog) * x + f.phylog
nl <- nrow(df)
if (is.null(y))
    y <- 1/2/nl + ((nl - 1):0)/nl
else y <- 1/2/nl + (nl - y)/nl
par(new = TRUE)
plot.default(0, 0, type = "n", xlab = "", ylab = "", xaxt = "n",
    yaxt = "n", xlim = c(-0.2, 1), ylim = c(0, 1), xaxs = "i",
    yaxs = "i", frame.plot = FALSE)
if (cexrow > 0) {
    for (i in 1:length(y)) {
        text(1.01, y[i], labels.row[i], adj = 0, cex = cexrow,
            xpd = NA)
        segments(1, y[i], 1.01, y[i], xpd = NA)
    }
}
if (cexcol > 0) {
    par(srt = 90)
    for (i in 1:length(x)) {
        if(col.col[i] == 0)
        text(x[i], 1.01, labels.col[i], adj = 0, cex = cexcol,
            xpd = NA, col = "grey")
            else
            text(x[i], 1.01, labels.col[i], adj = 0, cex = cexcol,
                xpd = NA, col = col.col[i])
        segments(x[i], 1, x[i], 1.01, , xpd = NA)
    }
    par(srt = 0)
}
if (grid) {
    col <- "lightgray"
    for (i in 1:length(y)) segments(1, y[i], f.phylog, y[i],
        col = col)
    for (i in 1:length(x)) segments(x[i], 0, x[i], 1, col = col)
}
rect(f.phylog, 0, 1, 1)
xtot <- x[col(as.matrix(df))]
ytot <- y[row(as.matrix(df))]
coeff <- diff(range(xtot))/15
z <- unlist(df)
sq <- sqrt(abs(z))
w1 <- max(sq)
sq <- csize * coeff * sq/w1
for (i in 1:length(z)) {
    if (sign(z[i]) >= 0) {
        if (fac.fg[i] == 0)
        symbols(xtot[i], ytot[i], squares = sq[i], bg = "black",
            fg = "grey", add = TRUE, inch = FALSE)
            else
            symbols(xtot[i], ytot[i], squares = sq[i], bg = "black",
                fg = fac.fg[i], add = TRUE, inch = FALSE)
    }
    else {
        symbols(xtot[i], ytot[i], squares = sq[i], bg = "white",
            fg = "grey", add = TRUE, inch = FALSE)
    }
}
br0 <- pretty(z, 4)
l0 <- length(br0)
br0 <- (br0[1:(l0 - 1)] + br0[2:l0])/2
sq0 <- sqrt(abs(br0))
sq0 <- csize * coeff * sq0/w1
sig0 <- sign(br0)
```

```
    dis <- phylog$droot
    dl <- phylog$droot[leave.names]
    dn <- phylog$droot[node.names]
    names(y) <- leave.names
    x <- dis
    x <- (x/max(x)) * f.phylog
    for (i in 1:n.leave) {
        segments(f.phylog, y[i], x[i], y[i], col = grey(0.7))
        points(x[i], y[i], pch = 20, cex = par("cex") * cleaves)
    }
    newlab <- as.character(1:length(phylog$nodes))
    newx <- NULL
    newy <- NULL
    yn <- rep(0, length(dn))
    names(yn) <- names(dn)
    y <- c(y, yn)
    for (i in 1:n.node) {
        w <- phylog$parts[[i]]
        if (clabel.nod > 0)
            newlab[i] <- labels.nod[i]
        but <- names(phylog$parts)[i]
        y[but] <- mean(y[w])
        newy[i] <- y[but]
        newx[i] <- x[but]
        b <- range(y[w])
        segments(x[but], b[1], x[but], b[2])
        x1 <- x[w]
        y1 <- y[w]
        x2 <- rep(x[but], length(w))
        segments(x1, y1, x2, y1)
    }
    if (cnodes > 0)
        points(newx, newy, pch = 21, bg = "white", cex = par("cex") *
            cnodes, xpd = NA)
    if (clabel.nod > 0)
        (scatterutil.eti(newx, newy, newlab, clabel.nod))
    if (clegend > 0)
        scatterutil.legend.bw.square(br0, sq0, sig0, clegend)
}
```

24. Classe d'objet pour l'utilisation des taxonomies

alias

as.taxo

description

'as.taxo' vérifie qu'un data.frame contient une taxonomie et lui confère la classe 'taxo', sous-classe de 'data.frame'. Le data frame ne doit comporter que des facteurs. Le facteur j doit définir des classes contenues entièrement dans les classes du facteur j-1. Typiquement, les lignes du tableau sont des espèces, les variables du tableau sont les genres, familles, ordres, classes, ….

usage

```
as.taxo(df)
```

arguments

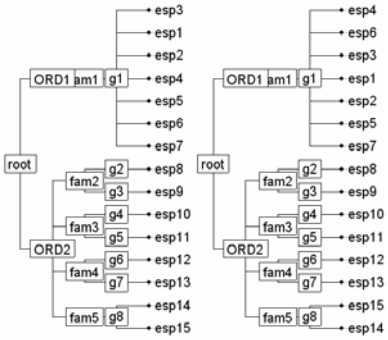
- df : est un objet de la classe 'data frame'

valeurs

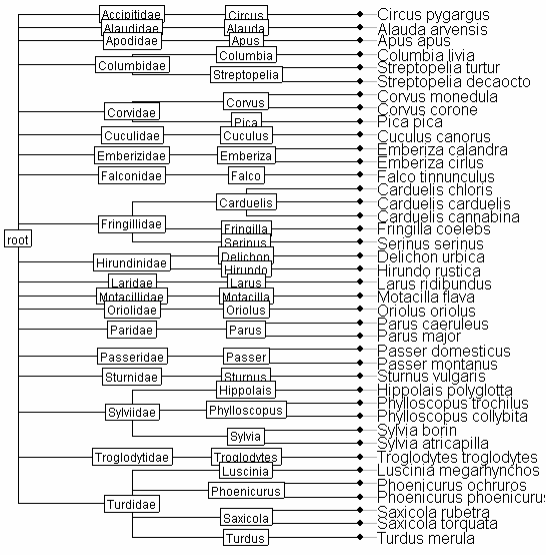
'as.taxo' renvoie un data.frame de classe 'taxo'. Les lignes sont rangées par ordre alphabétique pour chaque niveau.

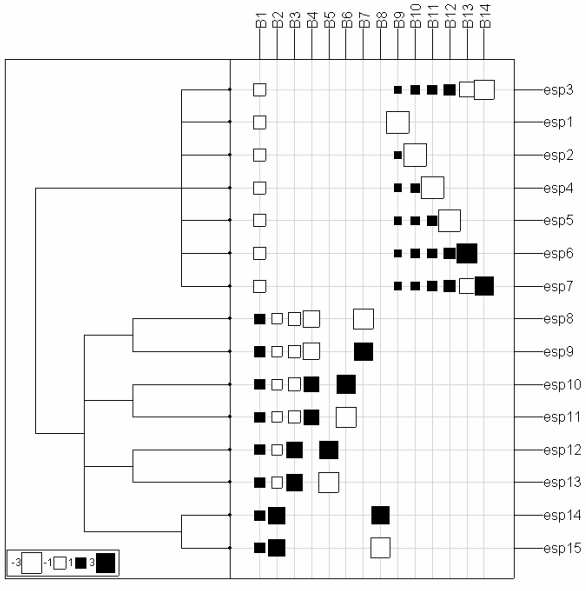
exemples

```
tax <- as.taxo(taxo.eg[[1]])
tax.phy <- taxo2phylog(as.taxo(taxo.eg[[1]]))
par(mfrow=c(1,2))
plot.phylog(tax.phy, clabel.l=1.25,
 clabel.n=1.25,f=0.75)
plot.phylog(taxo2phylog(
 as.taxo(taxo.eg[[1]][sample(15),])),
 clabel.l=1.25, clabel.n=1.25,f=0.75)
```




```
par(mfrow = c(1,1))
plot.phylog(taxo2phylog(
 as.taxo(taxo.eg[[2]])),clabel.l=1,
clabel.n=0.75,f=0.65)

tax.phy <- taxo2phylog(as.taxo(taxo.eg[[1]]))
table.phylog(tax.phy$Bscores,tax.phy,csi=1.5)
```

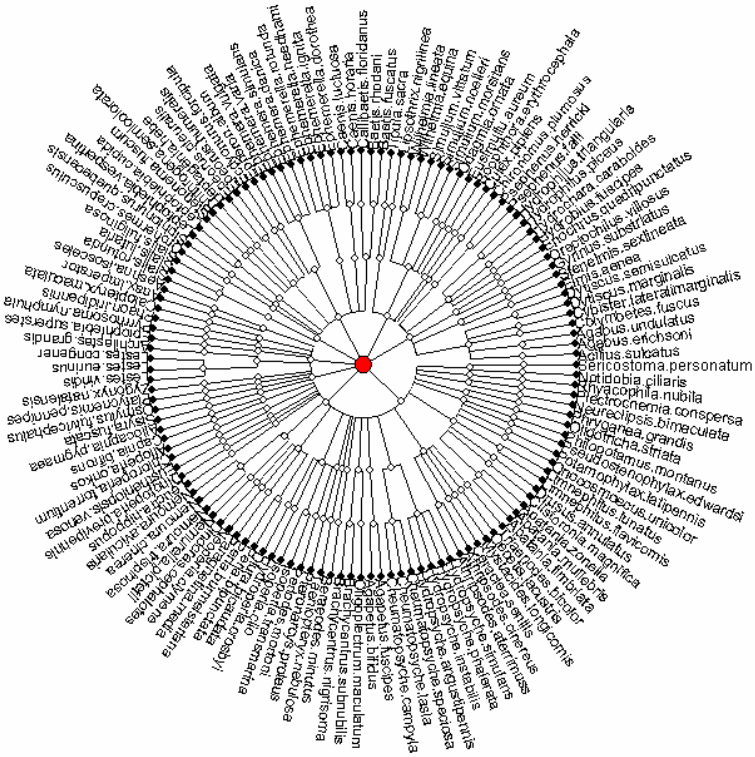




```
bsetal97$taxo[c(1:10,126:131),]
```

```
#                                    gen              fam         ord
# Acilius.sulcatus                Acilius        Dytiscidae  COLEOPTERA
# Agabus.erichsoni                 Agabus        Dytiscidae  COLEOPTERA
# Agabus.undulatus                 Agabus        Dytiscidae  COLEOPTERA
# Colymbetes.fuscus             Colymbetes        Dytiscidae  COLEOPTERA
# Cybister.lateralimarginalis     Cybister        Dytiscidae  COLEOPTERA
# Dytiscus.marginalis             Dytiscus        Dytiscidae  COLEOPTERA
# Dytiscus.semisulcatus           Dytiscus        Dytiscidae  COLEOPTERA
# Elmis.aenea                        Elmis           Elmidae  COLEOPTERA
# Stenelmis.sexlineata            Stenelmis           Elmidae  COLEOPTERA
# Gyrinus.substriatus             Gyrinus         Gyrinidae  COLEOPTERA
# …
# Phryganea.grandis              Phryganea      Phryganeidae TRICHOPTERA
# Neureclipsis.bimaculata     Neureclipsis Polycentropodidae TRICHOPTERA
# Plectrocnemia.conspersa     Plectrocnemia Polycentropodidae TRICHOPTERA
# Rhyacophila.nubila            Rhyacophila    Rhyacophilidae TRICHOPTERA
# Notidobia.ciliaris              Notidobia  Sericostomatidae TRICHOPTERA
# Sericostoma.personatum       Sericostoma  Sericostomatidae TRICHOPTERA
```

```
bsetal.phy <- taxo2phylog(as.taxo(bsetal97$taxo[,1:3]),F)
radial.phylog(bsetal.phy,clabel.l=0.75,cleaves=1,cnodes =1)
```



R code

```
as.taxo" <-
function (df)
{
    if (!inherits(df, "data.frame"))
        stop("df is not a data.frame")
    nr <- nrow(df)
    nc <- ncol(df)
    for (i in 1:nc) if (!is.factor(df[, i]))
        stop(paste("column", i, "of 'df' is not a factor"))
    for (i in 1:(nc - 1)) {
        t <- table(df[, c(i, i + 1)])
        w <- apply(t, 1, function(x) sum(x != 0))
        if (any(w != 1)) {
            print(w)
            stop(paste("non hierarchical design", i, "in", i +
                1))
        }
    }
    fac <- df[, nc]
    for (i in (nc - 1):1) fac <- fac:df[, i]
    df <- df[order(fac), ]
    class(df) <- c("data.frame", "taxo")
    return(df)
}
```

25. Décomposition de la variance par une famille de *k* formes bilinéaires symétriques

alias

val.kfbs

description

'val.kfbs' calcule les formes quadratiques $\mathbf{x}^t\mathbf{A}_k\mathbf{x}$ des variables d'un tableau avec trois types de normalisation (norme euclidienne : EU, norme spectrale : VP, norme de Smouse et Peakall : SP). Les formes quadratiques donnent une mesure de la structure d'une variable à différentes échelles.

usage

```
val.kfbs(tab, kfbs)
```

arguments

- *tab* : un tableau de données avec en colonne les variables dont on veut étudier la structure
- *kfbs* : un objet de la classe 'kfbs'

détails

Trois méthodes sont utilisées pour calculer les vecteurs $\mathbf{x}^t\mathbf{A}_k\mathbf{x}$. La première utilise la norme euclidienne $\frac{\mathbf{x}^t\mathbf{A}_k\mathbf{x}}{\sqrt{trace\left(\mathbf{A}_k^2\right)}}$. La seconde utilise la norme spectrale qui est majorée par la norme euclidienne $\frac{\mathbf{x}^t\mathbf{A}_k\mathbf{x}}{\sqrt{\lambda_1\left(\mathbf{A}_k^2\right)}}$. La troisième méthode utilise la norme introduite par Smouse et Peakall $\frac{\mathbf{x}^t\mathbf{A}_k\mathbf{x}}{\mathbf{x}^t\tilde{\mathbf{A}}_k\mathbf{x}}$.

A partir des familles dérivées des fonctions 'ttlv.kfbs' et 'knb2kfbs', on obtient des cousins du variogram et du correlogram suivant que l'argument 'method' prend la valeur Geary ou Moran.
A partir des familles dérivées des fonctions 'msbs.kfbs' et 'orthobasis2kfbs', on obtient des cousins du périodogramme défini par l'analyse spectrale et du scalogramme défini par les ondelettes.

valeurs

'val.kfbs' retourne une liste de 3 data frame, chacun étant associé à un type de normalisation. Chaque ligne d'un data frame correspond à la structure d'une variable. Chaque colonne correspond aux valeurs d'une des k formes quadratiques.

références

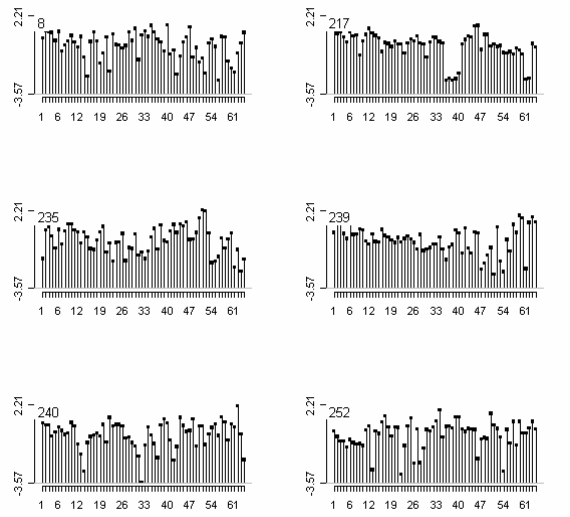
Chatelin, F. (1988) Valeurs propres de matrices Masson, Paris.
Smouse, P. & Peakall, R. (1999) Spatial autocorrelation analysis of individual multiallele and multilocus genetic structure. Heredity, 82, 561-573.

voir également

val.kfbs.rtest

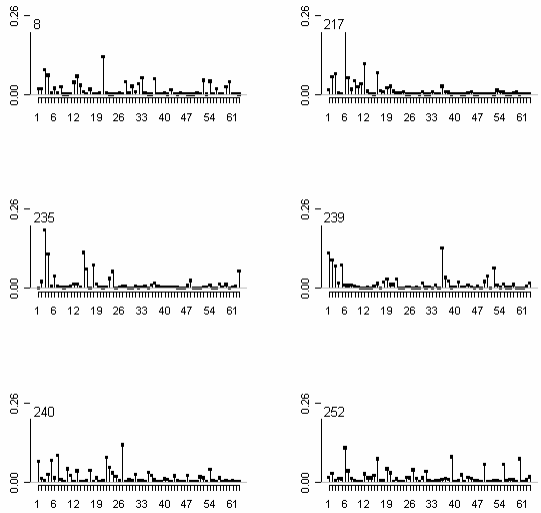
exemples

```
tab <- laser$laser[c(8, 215, 233,
                     237, 238, 250),]
tab <- as.data.frame(t(tab))
dotchart.line(tab, scaling = TRUE,
 ranging = TRUE, csub = 2,
 yjoining = -3.57)
```

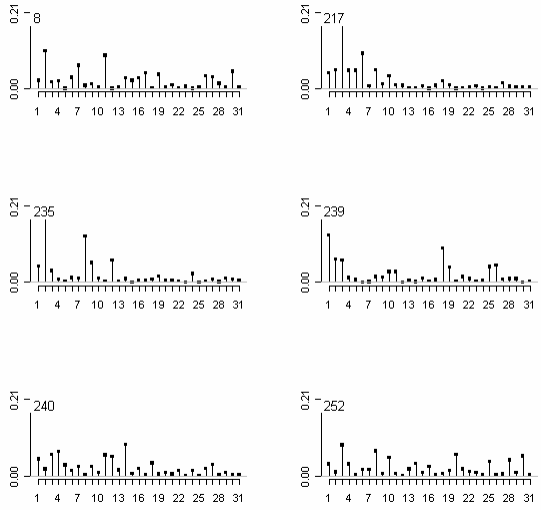



```
# orthogram avec la base associée
# à l'analyse spectrale
orthobas <- orthobasis.circ(64)
```

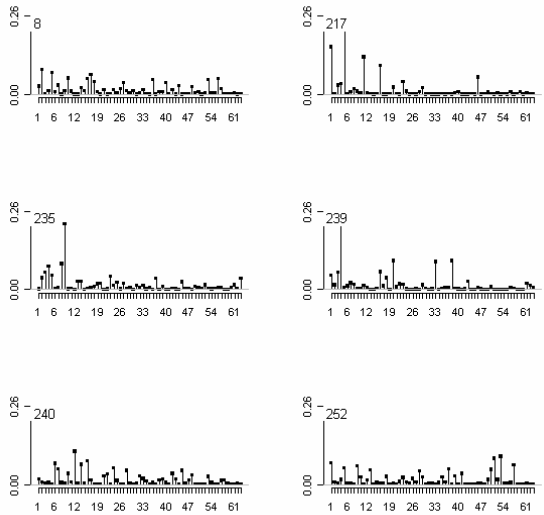
```
level <- as.factor(1:63)
kfbs <- orthobasis2kfbs(orthobas, level)
scalogram <- val.kfbs(tab, kfbs)$vvp
dotchart.line(as.data.frame(t(scalogram)),
 yjoining = 0, scaling = F, csub = 2)

val.kfbs.rtest(tab[,2], kfbs, 999)$vvp
class: krandtest
test number:   63
permutation number:   999
   test obs   P(X<=obs) P(X>=obs)
1  1    0.015 0.692     0.31
2  2    0.058 0.953     0.049
3  3    0.068 0.965     0.037
4  4    0.004 0.381     0.621
5  5    0.003 0.369     0.633
6  6    0.256 1         0.001
7  7    0.054 0.927     0.075
...
63 63   0.002 0.263     0.739
```



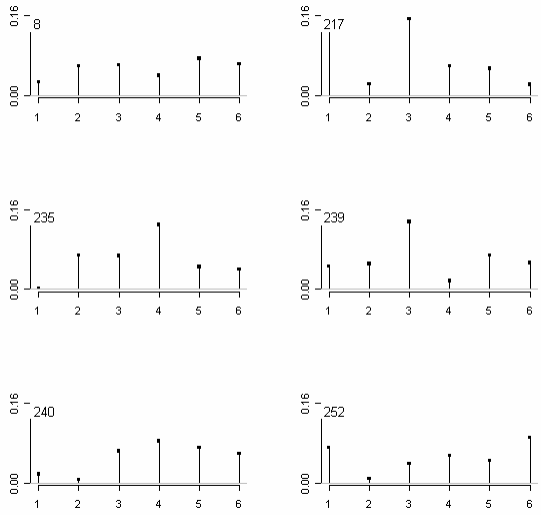

```
# périodogramme
level <- circ2level(63)
kfbs <- orthobasis2kfbs(orthobas, level)
periodogram <- val.kfbs(tab, kfbs)$veu
dotchart.line(as.data.frame(t(periodogram)),
 yjoining = 0, scaling = F, csub = 2)

val.kfbs.rtest(tab[,2], kfbs, 999)$veu
class: krandtest
test number:   31
permutation number:   999
   test obs   P(X<=obs) P(X>=obs)
1  1    0.044 0.811     0.191
2  2    0.051 0.909     0.093
3  3    0.183 1         0.001
4  4    0.05  0.9       0.102
...
30 30   0.005 0.179     0.823
31 31   0.003 0.106     0.896
```



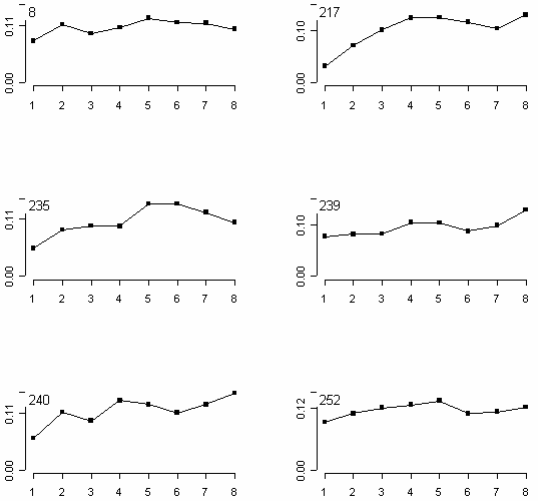

```
# scalogramme avec la base associée
# à l'ondelette de Haar
orthobas <- orthobasis.wavelet(64, "haar")
level <- as.factor(1:63)
kfbs <- orthobasis2kfbs(orthobas, level)
scalogram <- val.kfbs(tab, kfbs)$vvp
dotchart.line(as.data.frame(t(scalogram)),
 yjoining = 0, scaling = F, csub = 2)

val.kfbs.rtest(tab[,2], kfbs, 999)$vvp
class: krandtest
test number:   63
permutation number:   999
   test obs   P(X<=obs) P(X>=obs)
1  1    0.158 1         0.002
2  2    0.002 0.23      0.772
3  3    0.031 0.84      0.162
4  4    0.035 0.858     0.144
5  5    0.26  1         0.001
6  6    0.002 0.227     0.775
...
63 63   0.002 0.342     0.661

# waveletgram
```



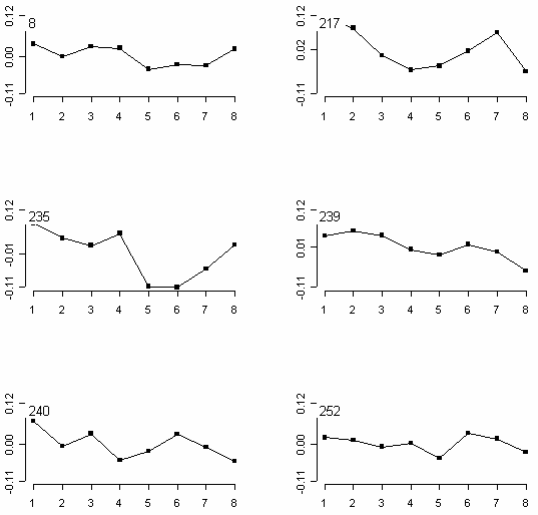
```
level <- wavelet2level(64)
kfbs <- orthobasis2kfbs(orthobas, level)
waveletgram <- val.kfbs(tab, kfbs)$veu
dotchart.line(as.data.frame(t(waveletgram)),
 yjoining = 0, scaling = F, csub = 2)
```




```
val.kfbs.rtest(tab[,2], kfbs, 999)$veu
class: krandtest
test number:    6
permutation number:    999
  test obs    P(X<=obs) P(X>=obs)
1 B6    0.158 0.999     0.003
2 B5    0.023 0.63      0.372
3 B4    0.153 1         0.001
4 B3    0.059 0.762     0.24
5 B2    0.054 0.32      0.682
6 B1    0.022 0.001     1

# variogramme
xy <- cbind.data.frame(x = 1:64,
 y = rep(1, 64))
knb <- bornes2knb(xy, d = c(0.5,
 1.5, 3.5, 5.5, 10.5, 15.5, 20.5,
 25.5, 30.5))
kfbs <- knb2kfbs(knb, "Geary")
variogram <- val.kfbs(tab, kfbs)$veu
dotchart.line(as.data.frame(t(variogram)),
 joining = FALSE, scaling = F,
 yranging = c(0, 0.15), cdot = 1,
 csub = 2)
```

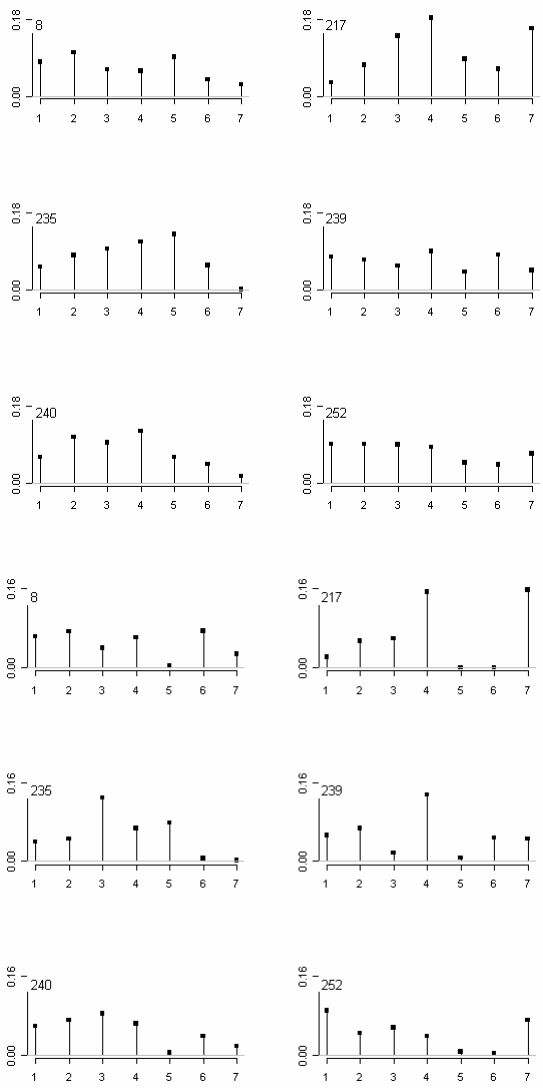



```
val.kfbs.rtest(tab[,2], kfbs, 999)$veu
class: krandtest
test number:    8
permutation number:    999
  test obs    P(X<=obs) P(X>=obs)
1 G1    0.032 0.001     1
2 G2    0.072 0.001     1
3 G3    0.101 0.186     0.816
4 G4    0.125 0.787     0.215
5 G5    0.125 0.87      0.132
6 G6    0.117 0.608     0.394
7 G7    0.104 0.202     0.8
8 G8    0.131 0.924     0.078

# correlogramme
kfbs <- knb2kfbs(knb, "Moran")
correlogram <- val.kfbs(tab, kfbs)$veu
dotchart.line(as.data.frame(t(correlogram)),
 joining = FALSE, scaling = F, cdot = 1,
 csub = 2)
```




```
val.kfbs.rtest(tab[,2], kfbs, 999)$veu
class: krandtest
test number:    8
permutation number:    999
  test obs     P(X<=obs) P(X>=obs)
1 M1    0.12    1         0.001
2 M2    0.083   1         0.001
3 M3    0.003   0.641     0.361
4 M4    -0.04   0.058     0.944
5 M5    -0.028  0.13      0.872
6 M6    0.016   0.871     0.131
7 M7    0.07    1         0.001
8 M8    -0.044  0.025     0.977

# analyse de Hill
kfbs <- ttlv.kfbs(64, c(1,2,4,8,16,24,32),
```

```
 method = "Geary")
hillgram <- val.kfbs(tab, kfbs)$veu
dotchart.line(as.data.frame(t(hillgram)),
 yjoining = 0, scaling = F, cdot = 1,
 csub = 2)

val.kfbs.rtest(tab[,2], kfbs, 999)$veu
class: krandtest
test number:   7
permutation number:   999
  test   obs   P(X<=obs) P(X>=obs)
1 ttg_1  0.155 0.001     1
2 ttg_2  0.318 0.075     0.927
3 ttg_4  0.414 0.999     0.003
4 ttg_8  0.363 1         0.001
5 ttg_16 0.117 0.974     0.028
6 ttg_24 0.067 0.942     0.06
7 ttg_32 0.158 1         0.002

# analyse de Noy-Meir
kfbs <- msbs.kfbs(64, c(1,2,4,8,16,24,32,64))
noygram <- val.kfbs(tab, kfbs)$veu
dotchart.line(as.data.frame(t(noygram)),
 yjoining = 0, scaling = F, cdot = 1,
 csub = 2)

val.kfbs.rtest(tab[,2], kfbs, 999)$veu
class: krandtest
test number:   7
permutation number:   999
  test   obs   P(X<=obs) P(X>=obs)
1 1_2    0.022 0.002     1
2 2_4    0.054 0.319     0.683
3 4_8    0.059 0.763     0.239
4 8_16   0.153 1         0.001
5 16_24  0     0.012     0.99  # pas de sens
6 24_32  0     0.001     1   # pas de sens
7 32_64  0.158 1         0.002
```



R code

```
"val.kfbs" <- function(kfbs, tab) {
    if (class(kfbs) != "kfbs") stop ("Non convenient data")

n <- attr(kfbs, "npoints")
nmet <- attr(kfbs, "nforms")
nommet <- attr(kfbs, "labels")
tr2 <- sqrt(attr(kfbs, "trace2"))
norm <- attr(kfbs, "norm")
nomvar <- names(tab)
tab <- as.matrix(tab)
tab <- scale(tab, center = TRUE, scale = FALSE)

    if (nrow(tab) !=n ) stop ('Non matched dim')

nvar <- ncol(tab)
veu <- data.frame(matrix(0, nvar, nmet), row.names = nomvar)
vsp <- veu
vvp <- veu
Jut <- function(x, A) sum(x*(A%*%x))
norm2x <- apply(tab, 2, function(x) sum(x*x))

decodemi <- function(A, tol =1e-07) {
    if (!is.matrix(A)) stop ("Matrix expected")
    if (sum( (A-t(A))^2 )>tol ) stop("Symetric matrix expected")
    A <- A-diag(A)/2
```

```
    A[col(A)<row(A)] <- 0
    sdv0 <- svd(A)
    l0 <- sdv0$d
    A <- sdv0$u%*%diag(l0)%*%t(sdv0$u)
    A <- A+sdv0$v%*%diag(l0)%*%t(sdv0$v)
    return(A)
}

for (i in 1:nmet) {
    a <- as.matrix.kfbs(kfbs, i)
    veu[, i] <- apply(tab, 2, Jut, A = a)
    b <- decodemi(a)
    vsp[, i] <- veu[, i]/apply(tab, 2, Jut, A = b)
    vvp[, i] <- veu[, i]/norm2x/norm[i]
    veu[, i] <- veu[, i]/norm2x/tr2[i]
}

names(veu) <- nommet
names(vsp) <- nommet
names(vvp) <- nommet
res <- list(veu = veu, vvp = vvp, vsp = vsp)
attr(res, "call") <- match.call()
return(res)
}
```

26. Tester l'absence de structure à différentes échelles

alias

val.kfbs.rtest

description

'val.kfbs.rtest' met en œuvre *k* tests de randomisation basés sur les métriques de structures.

usage

```
val.kfbs.rtest(x, kfbs, nrepet = 99)
```

arguments

- *x* : un vecteur correspondant à la variable dont on veut tester l'absence de structure à différentes échelles
- *kfbs* : un objet de la classe 'kfbs'
- *nrepet* : un entier représentant le nombre de permutations utilisées pour construite les enveloppes de confiance

valeurs

'val.kfbs.rtest' retourne un objet de la classe 'krandtest'

voir également

val.kfbs

exemples

```
# voir la fonction val.kfbs(…) (Annexe 2.25)
```

R code

```
"val.kfbs.rtest" <- function(x, kfbs, nrepet = 99){
    if(!inherits(kfbs,"kfbs")) stop ("object of class 'kfbs' expected")

n <- length(x)
tab <- apply(as.data.frame(rep(n, nrepet)), 1, sample)
tab <- cbind.data.frame(1:n, tab)
tab <- apply(tab, 2, function(u) x[u])
res <- val.kfbs(tab, kfbs)
res <- lapply(res, as.list)
class(res$veu) <- "krandtest"
class(res$vvp) <- "krandtest"
class(res$vsp) <- "krandtest"
return(res)
}
```

LES ARTICLES

1. Taking into account spatial dependence in multivariate analysis: a generalization of Wartenberg's multivariate spatial correlation

auteurs

Sébastien Ollier
Stéphane Dray
Daniel Chessel

résumé

Identification and measurement of spatial autocorrelation is one topic of great interest in applied geography. Although univariate measurement such as Moran's *I* and Geary's *c* are well known since a long time, extensions to the multivariate case are rare. Here, we propose a multivariate spatial analysis based on Moran's *I* (MULTISPATI) by introducing a row-sum standardized spatial weight matrix in the statistical triplet notation. MULTISPATI approach is very flexible and can handle various kinds of data (quantitative and/or qualitative data, contingency tables…). This analysis, which is a generalization of Wartenberg's approach to multivariate spatial correlation, would find a compromise between the relations among many variables (multivariate analysis) and their spatial structure (autocorrelation). We propose a permutation test based on this analysis in order to measure the statistical significance of a multivariate spatial structure. We illustrate the method using the Irish counties data set of Geary (1954).

journal

Geographical Analysis: non accepté, en révision

Introduction

The identification and measurement of the spatial component of a variable have been a major issue in geographical analysis. Global indices such as Moran's *I* and Geary's *c* (Cliff and Ord 1973, Geary 1954, Moran 1948) and their local extensions (Anselin 1995) have been widely used to measure the spatial dependence and its local variations for one quantitative variable. Indices for nominal data are also well known (Krishna Iyer 1949). All these approaches focus on the spatial structure of observations considering only one single variable.

In the case of more than one variable, multivariate analysis is a natural tool to summarize large data sets. Various existing methods allow to deal with quantitative data (Principal Component Analysis, PCA, Hotelling 1933), qualitative data (Multiple Correspondence Analysis, MCA, Tenenhaus and Young 1985), contingency tables (Correspondence Analysis, CA, Greenacre 1984) or mixture of quantitative and qualitative data (PCAMIX, Kiers 1994)… Classical multivariate techniques are often used on spatial data and often usefully. The usual approach consists in firstly performing multivariate analysis to summarize the data and secondly plotting the scores in the geographical space in order to represent the spatial component of the structures identified (e.g., Dray *et al.* 2003c, Goodall 1954, Kadmon and Danin 1997). However, classical multivariate analyses do not take into account space in their computation and are not specifically oriented to the identification of spatial structure.

As spatial processes could originate the observed structures, it seems essential to include the spatial dependence of observations in the analysis of multivariate georeferenced data. Spatial multivariate techniques, i.e. methods of multivariate analysis devoted to the identification of spatial structure (e.g. spatial patches, regional trends), are relatively undeveloped although this problematic is encountered across a wide range of fields. The first interesting attempt that aims at depicting basic spatial patterns is due to Wartenberg (1985). Recently, Lee (2001) showed that Wartenberg's approach has major drawbacks and proposed a bivariate spatial association measure which can be easily used for spatial multivariate analysis. Other methods (see review in Bailey and Krzanowski 2000) have been developed in various fields such as spatial imagery (Switzer and Green 1984), or geosciences (Grunsky and Agterberg 1991). As Wartenberg's method, all these approaches consist in the diagonalisation of a spatial covariance or correlation matrix to identify multivariate spatial association and are restricted to the case of quantitative (normalized) variables.

In this paper, we propose a new method of spatial multivariate analysis. Contrary to previously cited methods that deal only with (normalized) quantitative data, our approach is very general. It introduces spatial constraint in classical multivariate methods and allows, for example, to perform spatial analysis of contingency tables or mixture of quantitative and qualitative variables. It can be seen as a generalization of Wartenberg's approach taking into account the reproaches of Lee (2001). We firstly introduce some simple elements of spatial analysis and secondly we present the general framework of multivariate analysis using the statistical triplet notation. Then, the principles of a new spatial multivariate analysis are given and the method is illustrated on a real data set.

Measurements of spatial association

Geary's *c* and Moran's *I*

Let us consider **x**, a vector composed by the measurements of a variable for *n* spatial units, i.e. $\mathbf{x}^t = (x_1, ..., x_n)$. The Moran's *I* is given by:

$$I(\mathbf{x}) = \frac{n \sum_{(2)} c_{ij} (x_i - \bar{x})(x_j - \bar{x})}{\sum_{(2)} c_{ij} \sum_{i=1}^{n} (x_i - \bar{x})^2}$$

(1)

and the Geary's *c* is :

$$c(\mathbf{x}) = \frac{(n-1) \sum_{(2)} c_{ij} (x_i - x_j)^2}{2 \sum_{(2)} c_{ij} \sum_{i=1}^{n} (x_i - \bar{x})^2}$$

(2)

where $\sum_{(2)} = \sum_{i=1}^{n} \sum_{j=1}^{n}$ with $i \neq j$ and $\mathbf{C} = [c_{ij}]$ is a spatial connectivity matrix.

Geary's *c* is always positive while Moran's *I* can be positive or negative. The two indices are rewritten using matrix formulation. For Moran's *I*, we obtain:

$$I(\mathbf{x}) = \frac{n}{\mathbf{1}^t \mathbf{C} \mathbf{1}} \frac{\mathbf{x}^t \mathbf{N} \mathbf{C} \mathbf{N} \mathbf{x}}{\mathbf{x}^t \mathbf{N} \mathbf{x}} = \frac{n}{\mathbf{1}^t \mathbf{C} \mathbf{1}} \frac{\mathbf{z}^t \mathbf{C} \mathbf{z}}{\mathbf{z}^t \mathbf{z}}$$

(3)

where $\mathbf{N} = (\mathbf{I} - (1/n)\mathbf{1}\mathbf{1}^t)$ is a centering operator, $\mathbf{1}^t = (1, ..., 1)$ a (1 by *n*) row vector, $\mathbf{I}$ the identity matrix and $\mathbf{z} = \mathbf{N}\mathbf{x}$ the centered variable. If $\mathbf{C}$ is symmetric, Geary's *c* is equal to (Lebart 1969):

$$c(\mathbf{x}) = \frac{(n-1)}{\mathbf{1}^t\mathbf{C}\mathbf{1}} \frac{\mathbf{x}^t(\mathbf{D_C} - \mathbf{C})\mathbf{x}}{\mathbf{x}^t\mathbf{N}\mathbf{x}} = \frac{(n-1)}{\mathbf{1}^t\mathbf{C}\mathbf{1}} \frac{\mathbf{z}^t(\mathbf{D_C} - \mathbf{C})\mathbf{z}}{\mathbf{z}^t\mathbf{z}}$$

(4)

with $\mathbf{D_C} = diag(\sum_{j=1}^{n} c_{ij}) = diag(\sum_{i=1}^{n} c_{ij})$.

The choice of weights

The matrix $\mathbf{C} = \left[c_{ij} \right]$ is a weighting matrix (Bavaud 1998, Tiefelsdorf *et al.* 1999) which indicates the strength of the potential interaction between spatial units. In the Cliff and Ord's book (1973), it is specified that "*the use of a generalised weighting matrix [...] allows the investigator to choose a set of weights which he deems appropriate from prior considerations. This allows great flexibility*". The binary connectivity version of **C** (**B**) whose elements b_{ij} equal 1 for contiguous spatial units and 0 otherwise is often used. Econometricians prefer to use the row-sum standardized version of **C** ($\mathbf{W} = \left[c_{ij} / \sum_{j=1}^{n} c_{ij} \right]$) which allows easier interpretation of autoregressive models (Ord 1975). Doubly standardized spatial weights matrix is also often used with sum of all elements equal to 1 ($\mathbf{F} = \left[c_{ij} / \sum_{(2)} c_{ij} \right]$) or to *n* ($n\mathbf{F}$). de Jong *et al.* (1984) provide exact lower and upper bounds for *c* and *I* for a given connection matrix. These extremes are given by the smallest and largest eigenvalues of $\mathbf{NBN}$ and $\mathbf{N}(\mathbf{W} + \mathbf{W}^t)\mathbf{N}/2$, for the **B** and **W** weighting options respectively, while the eigenvectors of these matrices (Griffith 1996, Griffith 2000a) can be used for spatial filtering purposes (Getis and Griffith 2002, Griffith 2000b).

The choice of the spatial weight matrix is the most critical step in computing a measure of spatial association because it can influence the significance of the test (Tiefelsdorf *et al.* 1999). Moreover, it defines also the limits of autocorrelation measures.

When **W** is applied, Lee (2001) proposes a nice decomposition of Moran's *I* into two parts using the concept of spatial lag (Anselin 1996). The lag vector is composed of the averages of neighbors weighted by the spatial connection matrix and is computed by:

$$\tilde{\mathbf{x}} = \mathbf{W}\mathbf{x} \text{ (i.e. } \tilde{x}_i = \sum_{j=1}^{n} w_{ij} x_j \text{)}$$

(5)

Anselin (1996) proposes to study spatial autocorrelation with Moran scatterplot by plotting the original variable (**x**) against the spatial lag of the variable ($\tilde{\mathbf{x}}$). The use of the weighting matrix **W** reduces (3) to:

$$I(\mathbf{x}) = \frac{\mathbf{x}^t\mathbf{NWNx}}{\mathbf{x}^t\mathbf{Nx}} = \frac{\mathbf{z}^t\mathbf{Wz}}{\mathbf{z}^t\mathbf{z}} = \frac{\mathbf{z}^t\tilde{\mathbf{z}}}{\mathbf{z}^t\mathbf{z}}$$

(6)

Row-sum standardization implies that Moran's *I* is reduced to a ratio of quadratic forms which can be easily interpreted and it provides a sort of smoothing operator (lag vector). In the case of regular lattice, the number of neighbors tends to constant and the use of **B** (or **F**) weights can be justified. One can think of a "spatial lag" as **Cz** in (3) as a sum of the neighboring values. If the number of neighbors is not constant, we have a problem with scale, where these values will be larger are a function of the number of neighbors. This may make sense in some contexts, but there are many where it doesn't. The row-standardization avoids this problem and creates a variable that is simply an average of the neighbors and thus will be comparable to the value of the original observations.

Moreover, it facilitates comparisons between spatial parameters in the case of spatial autoregressive model. The parameter space (the range of allowed values) is determined

by the values in the weights matrix. For a row-standardized weights matrix, the maximum is always 1. For an un-standardized weights matrix, the maximum depends on the values in the matrix: high weights yield small parameter values and vice versa. For example, in many of the earlier studies with rook (unstandardized) weights on a regular lattice the maximum value is 0.25. So when comparing studies, it could be that the 0.25 is actually "larger than" 0.5 for row-standardized weights (L. Anselin, *personal communication*).

All these considerations justify the preference for this weighting option to compute Moran's *I*. Another argument arises from the work of Lee (2001) whose rewrite Moran's *I* to develop a bivariate spatial association measure:

$$I(\mathbf{x}) = \frac{\sum_{i=1}^{n}(x_i-\overline{x})(\tilde{x}_i-\overline{x})}{\sum_{i=1}^{n}(x_i-\overline{x})^2} = \sqrt{\frac{\sum_{i=1}^{n}(\tilde{x}_i-\overline{x})^2}{\sum_{i=1}^{n}(x_i-\overline{x})^2}} \cdot \underbrace{\sqrt{\frac{\sum_{i=1}^{n}(\tilde{x}_i-\overline{\tilde{x}}_i)^2}{\sum_{i=1}^{n}(\tilde{x}_i-\overline{x})^2}}}_{\cong 1} \cdot r_{\mathbf{x},\tilde{\mathbf{x}}} \cong \sqrt{SSS_{\mathbf{x}}} \cdot r_{\mathbf{x},\tilde{\mathbf{x}}}$$

(7)

As the second part of the middle element is approximately 1, Moran's *I* can be regarded as the product of a spatial smoothing scalar (SSS) by the Pearson correlation between the variable and its spatial lag. Then, Lee suggests that a bivariate spatial association measure should include a "*point-to-point association between two variables, which requires the inclusion of a certain form of Pearson's correlation between the two variables*" and "*should reflect the degrees of spatial autocorrelation for both variables under investigation. In other*

words, it should respond to the collective effect of the SSSs of the variables". Unfortunately, his approach considers the correlation between the spatial lag vectors but not between the original variables and he found that these two quantities could have different signs.

In the next section, we present a new approach to spatial multivariate analysis by introducing the row-standardized weights matrix **W** in the context of classical multivariate analysis.

Classical multivariate analysis

Classical multivariate analysis is a natural tool to summarize large data sets. Various methods are available to take into account the different characteristics of the data (quantitative or qualitative variables, contingency tables…). For a review in ecology, the reader should consult Dray *et al.* (2003a). The notion of statistical triplet (Cailliez and Pagès 1976, Escoufier 1987) provides a theoretical framework and efficient way to define multivariate analyses.

Let (**X**, **Q**, **D**) be a statistical triplet where **X** is a $(n \times p)$ matrix derived from any data table, **D** be a scalar product of $\mathbb{R}^n$ (n by n symmetric matrix) and **Q** be scalar product of $\mathbb{R}^p$ (p by p symmetric matrix). The generalized singular value decomposition (GSVD, Greenacre 1984) of (**X**, **Q**, **D**) consists in finding the vectors $\mathbf{u}_1$ (first principal axis) and $\mathbf{v}_1$ (first principal component) so that the inner products,

$$\left(\mathbf{XQu}_1 \mid \mathbf{v}_1\right)_{\mathbf{D}} = \mathbf{u}_1{}^t\mathbf{QX}^t\mathbf{Dv}_1 = \mathbf{v}_1{}^t\mathbf{DXQu}_1 = \left(\mathbf{X}^t\mathbf{Dv}_1 \mid \mathbf{u}_1\right)_{\mathbf{Q}}$$

(8)

and such that the quadratic forms $Q(\mathbf{u}_1)$ and $S(\mathbf{v}_1)$,

$$\begin{aligned} Q(\mathbf{u}_1) &= \left\|\mathbf{XQu}_1\right\|_{\mathbf{D}}^2 = \mathbf{u}_1{}^t\mathbf{QX}^t\mathbf{DXQu}_1 \\ S\left(\mathbf{v}_1\right) &= \left\|\mathbf{X}^{\mathrm{T}}\mathbf{Dv}_1\right\|_{\mathbf{Q}}^2 = \mathbf{v}_1{}^t\mathbf{DXQX}^t\mathbf{Dv}_1 \end{aligned}$$

(9)

are maximized under the constraints that $\left\|\mathbf{u}_1\right\|_{\mathbf{Q}}^2 = \mathbf{u}_1{}^t\mathbf{Qu}_1 = \left\|\mathbf{v}_1\right\|_{\mathbf{D}}^2 = \mathbf{v}_1{}^t\mathbf{Dv}_1 = 1$.

The achieved maximum of the above inner product is equal to the first singular value. The solution vectors $\mathbf{u}_1$ and $\mathbf{v}_1$ can also be obtained as the right-hand eigenvectors of $\mathbf{X}^t\mathbf{DXQ}$ and $\mathbf{XQX}^t\mathbf{D}$, respectively, and the maxima of $Q(\mathbf{u}_1)$ and $S(\mathbf{v}_1)$ are equal and given by the first eigenvalue (λ_1) of these matrices (which is the square of the above-mentioned first singular value). Rows ($\mathbf{a}_1$) and columns ($\mathbf{b}_1$) scores are then computed by projection as follows:

$$\mathbf{a}_1 = \mathbf{XQu}_1 \text{ and } \mathbf{b}_1 = \mathbf{X}^t\mathbf{Dv}_1$$

(10)

If r is the rank of the matrix $\mathbf{X}$, then the second and further principal axes ($\mathbf{u}_2$, $\mathbf{u}_3$, …, $\mathbf{u}_r$) and the second and further principal components ($\mathbf{v}_2$, $\mathbf{v}_3$, …, $\mathbf{v}_r$) maximize the same inner products and norms but are subjected to extra constraints of orthogonality, i.e. for all $s \neq t$ $\left(\mathbf{u}_s \mid \mathbf{u}_t\right)_{\mathbf{Q}} = \left(\mathbf{v}_s \mid \mathbf{v}_t\right)_{\mathbf{D}} = 0$.

Various definitions of the matrices $\mathbf{X}$, $\mathbf{Q}$ and $\mathbf{D}$ correspond to different multivariate analysis of an original table $\mathbf{Y}$: centered PCA ($\mathbf{X} = \left[y_{ij} - \bar{y}_j \right], \mathbf{Q} = \mathbf{I}_p, \mathbf{D} = \frac{1}{n}\mathbf{I}_n$), normed PCA ($\mathbf{X} = \left[\frac{y_{ij} - \bar{y}_j}{\sigma_j} \right], \mathbf{Q} = \mathbf{I}_p, \mathbf{D} = \frac{1}{n}\mathbf{I}_n$), correspondence analysis ($\mathbf{X} = \left[\frac{y_{ij}}{y_{i+}y_{+j}} - 1 \right]$, $\mathbf{Q} = diag(\frac{y_{+1}}{y_{++}}, ..., \frac{y_{+p}}{y_{++}})$, $\mathbf{D} = diag(\frac{y_{1+}}{y_{++}}, ..., \frac{y_{n+}}{y_{++}})$) with $y_{i+} = \sum_{j=1}^{p} y_{ij}$, $y_{+j} = \sum_{i=1}^{n} y_{ij}$ and $y_{++} = \sum_{i=1}^{n}\sum_{j=1}^{p} y_{ij}$ …

A new spatial multivariate analysis

MULTISPATI Analysis

We present a new approach, the multivariate spatial analysis based on Moran's *I* (MULTISPATI), by introducing the row-sum standardized weight matrix $\mathbf{W}$ in the analysis of a statistical triplet ($\mathbf{X}$, $\mathbf{Q}$, $\mathbf{D}$). It is possible to extend the concept of lag vector to construct a lag matrix $\tilde{\mathbf{X}} = \mathbf{WX}$. The two tables $\tilde{\mathbf{X}} = \mathbf{WX}$ and $\mathbf{X}$ are fully matched, i.e. it contains the measurements of the same variables for the same sites. This situation can be treated by the

coinertia analysis (Dolédec and Chessel 1994, Dray *et al.* 2003a) of a pair of fully matched tables (Dray *et al.* 2003b, Torre and Chessel 1995). In the case of the two tables $\tilde{\mathbf{X}} = \mathbf{WX}$ and $\mathbf{X}$, it corresponds to the diagonalization of the statistical triplet $(\mathbf{X}, \mathbf{Q}, 1/2(\mathbf{W}^t\mathbf{D} + \mathbf{DW}))$ and the first eigenvalue is given by the maxima of (9).

$$\begin{aligned} Q(\mathbf{u}_1) &= \left\|\mathbf{XQu}_1\right\|^2_{1/2(\mathbf{L}^t\mathbf{W}+\mathbf{DW})} \\ &= \mathbf{u}_1{}^t\mathbf{Q}^t\mathbf{X}^t(1/2(\mathbf{W}^t\mathbf{D} + \mathbf{DW}))\mathbf{XQu}_1 \\ &= 1/2(\mathbf{u}_1{}^t\mathbf{Q}^t\mathbf{X}^t\mathbf{W}^t\mathbf{DXQu}_1 + \mathbf{u}_1{}^t\mathbf{Q}^t\mathbf{X}^t\mathbf{DWXQu}_1) \\ &= 1/2\left(\mathbf{XQu}_1 \mid \mathbf{WXQu}_1\right)_{\mathbf{D}} + 1/2\left(\mathbf{WXQu}_1 \mid \mathbf{XQu}_1\right)_{\mathbf{D}} \\ &= \mathbf{u}_1{}^t\mathbf{Q}^t\mathbf{X}^t\mathbf{DWXQu}_1 = \mathbf{a}_1{}^t\mathbf{DWa}_1 = \mathbf{a}_1{}^t\mathbf{D\tilde{a}}_1 \end{aligned}$$

(11)

This analysis maximizes the scalar product between a linear combination of original variables ($\mathbf{a}_1$) and a linear combination of lagged variables ($\tilde{\mathbf{a}}_1$). Equation (11) can be rewritten as:

$$Q(\mathbf{u}_1) = \frac{\mathbf{u}_1{}^t\mathbf{Q}^t\mathbf{X}^t\mathbf{DWXQu}_1}{\mathbf{u}_1{}^t\mathbf{Q}^t\mathbf{X}^t\mathbf{DXQu}_1}\mathbf{u}_1{}^t\mathbf{Q}^t\mathbf{X}^t\mathbf{DXQu}_1 = I\left(\mathbf{XQu}_1\right)\left\|\mathbf{XQu}_1\right\|^2_{\mathbf{D}} = I\left(\mathbf{a}_1\right)\left\|\mathbf{a}_1\right\|^2_{\mathbf{D}}$$

(12)

This formulation shows that MULTISPATI finds a linear combination of variables ($\mathbf{XQu}_1$) which maximizes a compromise between the classical multivariate analysis ($\left\|\mathbf{XQu}_1\right\|^2_{\mathbf{D}}$) and a generalized version of Moran's I ($I\left(\mathbf{XQu}_1\right)$). In practice, it is preferable to diagonalize the **Q**-symmetric matrix $\mathbf{H} = (1/2)(\mathbf{X}^t(\mathbf{W}^t\mathbf{D} + \mathbf{DW})\mathbf{XQ})$ instead of $\mathbf{X}^t\mathbf{DWXQ}$ which is not symmetric.

In the case of the normed PCA of an original table **Y** (see previous section), the elements in matrix **H** can be written as:

$$
\begin{aligned}
\mathbf{H}_{jk} &= (1/2)(\mathbf{x}_j^{\,t}(\mathbf{W}^t\mathbf{D}+\mathbf{D}\mathbf{W})\mathbf{x}_k\mathbf{Q}) \\
&= \frac{1}{2n}\sum_{i=1}^{n}\frac{\tilde{y}_{ij}-\bar{y}_j}{\sigma_j}\frac{y_{ik}-\bar{y}_k}{\sigma_k}+\frac{1}{2n}\sum_{i=1}^{n}\frac{y_{ij}-\bar{y}_j}{\sigma_j}\frac{\tilde{y}_{ik}-\bar{y}_k}{\sigma_k} \\
&= \frac{1}{2}\sum_{i=1}^{n}\frac{\tilde{y}_{ij}-\bar{y}_j}{\sqrt{\sum_{i=1}^{n}(y_{ij}-\bar{y}_j)^2}}\frac{y_{ik}-\bar{y}_k}{\sqrt{\sum_{i=1}^{n}(y_{ik}-\bar{y}_k)^2}} \\
&\quad+\frac{1}{2}\sum_{i=1}^{n}\frac{y_{ij}-\bar{y}_j}{\sqrt{\sum_{i=1}^{n}(y_{ij}-\bar{y}_j)^2}}\frac{\tilde{y}_{ik}-\bar{y}_k}{\sqrt{\sum_{i=1}^{n}(y_{ik}-\bar{y}_k)^2}} \\
&\cong \frac{1}{2}\sqrt{SSS_{\mathbf{y}_j}}\,r_{\tilde{\mathbf{y}}_j\mathbf{y}_k}+\frac{1}{2}\sqrt{SSS_{\mathbf{y}_k}}\,r_{\tilde{\mathbf{y}}_k\mathbf{y}_j}
\end{aligned}
$$

(13)

and using (7) and (13), it is easy to show that $\mathbf{H}_{kk} = I(\mathbf{y}_k)$. In this case, MULTISPATI is equivalent to Wartenberg's approach using a row-sum weighting scheme (more details are given in the discussion section).

The MULTISPATI approach (and the permutation test presented below) has been implemented in the R software (Ihaka and Gentleman 1996) as a function of the ade4 package to be used in connection with the spdep package.

Permutation test

In order to test the statistical significance of the spatial structure of the table **X**, we have develop a permutation test based on the multivariate spatial association measure defined above. This test is based on permutation of rows table **X** and for each permutation the total inertia of the analysis is computed (Kazi-Aoual *et al.* 1995). The total inertia is equal to $trace\left(\mathbf{X}^t\mathbf{D}\mathbf{W}\mathbf{X}\mathbf{Q}\right)$ and increases with the intensity of the link between **X** and $\tilde{\mathbf{X}} = \mathbf{W}\mathbf{X}$. We compare the observed value to those obtained by permutation in order to test the global spatial structure for the table **X**. We used a Monte-Carlo version of this test based on 9999 permutations.

Application

For the purpose of illustration, we reanalyze the famous Irish county data set (Geary 1954) which is available in the ade4 package. Twelve socio-economic variables have been measured for twenty-five counties (Dublin has been excluded). In order to illustrate the case of a mixture of quantitative and qualitative variables, we have slightly modify the original data set. In Geary (1954), valuation level was coded by three variables indicating the percentage number of agricultural holdings in valuation groups (<10 £, 10-50 £, >50 £). We

perform k-means clustering one these three variables to obtain a qualitative variable with three levels (high, medium and low valuation). We apply PCAMIX which is equivalent to MCA if there are only qualitative variables and to normed PCA if there are only quantitative variables. The principal components of this analysis are normalized vectors maximizing the sum of squared correlations with quantitative variables and of correlation ratios with qualitative variables. We then perform the MULTISPATI-PCAMIX by introducing the spatial constraint using a connectivity matrix based on common boundaries (i.e. the connectivity between two counties is equal the length of their common boundary).

Barplot of eigenvalues shows that the main structure can be summarized by one axis for PCAMIX and by two for MULTISPATI-PCAMIX (figure 1). It is well known that most of the variables of this data set are very autocorrelated and this strong spatial structure has two consequences on our results. Firstly, and not surprisingly, the permutation test is very significant (total inertia = 3.8976, p = 0.0001). Secondly, as the main structure is due to spatial processes, results from the standard analysis and the spatially constrained are quite similar. The first two principal axes of PCAMIX are highly correlated to those of MULTISPATI-PCAMIX (figure 2).

For the two analyses, the first axis highlights differences between densely-populated counties with high proportion of larger farms (higher level of taxation) and of cars and poor counties with high percentage of single man, low level of taxation and low retail sales (figure 3). The second axis distinguishes counties with many pigs and milch cows.

As the results are quite similar, one can have any doubt about the advantages of MULTISPATI-PCAMIX compared to PCAMIX in this case. However, if we plot on the factorial plane the scores and the lagged scores for each county, it is clear that the spatial constraint facilitates the interpretation (figure 4). While it is rather unclear for the classical approach (figure 4a), MULTISPATI-PCAMIX distinguishes three groups of counties (figure 4b). This is confirmed by a hierarchical clustering performed on the scores and lagged scores of counties for the first two axes (figure 4c). We obtain a partition of Ireland into three area: a northern part with a high percentage of single man, low level of taxation and low retail, a south-eastern part with high proportion of larger farms (higher level of taxation) and of cars and south-western part characterized by farms with milk cows and pigs. The North-South pattern is identified by the first axis (figure 5a) and the East-West by the second one (figure 5b). Due to the spatial constraint, these maps are more smoothed than those that we can obtain with classical analysis.

For a given axis, Moran's *I* for scores produced by the spatially constrained analysis are always superior to those obtained by the classical method (table 1). One can be surprised at seeing that the first axis of the spatial analysis has lower autocorrelation than the second one (table 1). This is due to the fact that eigenvalues of MULTISPATI are a compromise between the criteria of the classical method and the spatial constraint (equation 12). The first axis is slightly less autocorrelated than the second one (*I*=0.59 and *I*=0.64) but more structured (var=4.32 and var=1.77) and corresponds to higher eigenvalue (which is the product of the variance by the autocorrelation). The low autocorrelation obtained for axis 1 is probably due to Kerry (label H on figure 4c) which has very high score compared to its neighbors (figure 5). In our study, this county has only two neighbors and is not connected to Clare (label C on figure 4c). These two counties are very close in term of distances but they are separated by the sea (Shannon estuary) and have no common boundaries and so they are not considered as neighbors in the connectivity matrix. In his paper, Geary (1954) did not say how he defined the connectivity matrix but he produced a map from which we can deduce that Clare and Kerry are considered as neighbors. This is probably more realistic than our definition of the spatial connectivity and would probably increase autocorrelation for the first axis. The hypothesis about the low autocorrelation due to Kerry county, is confirmed by the figure 4b. On this figure, Kerry is represented by a very long horizontal arrow. This indicates a very low spatial correlation between these county and its neighbors. This can be interpret as a local index of spatial association (Anselin 1995).

Discussion

The major advantage of the MULTISPATI approach is its generality because it is simply based by the introduction of the spatial weight matrix in the statistical triplet notation. Hence, one can perform MULTISPATI-CA for contingency tables, MULTISPATI-MCA for qualitative data… As shown before (1), Moran's *I* is negative when negative autocorrelation appears and our approach, which is based on this index, will produce negative eigenvalues. In the case of high negative eigenvalue, it is important to have a look to the associated eigenvectors which can depict strong local structures of interest (negative autocorrelation). On the contrary, Geary's *c* is always positive (2) and that is probably why the first attempts to spatial multivariate analysis are based on this index using the Euclidean metrics $(\mathbf{D}_\mathbf{C} - \mathbf{C})$ (4). Following the initial work of Lebart (1969), many methods based on the metrics $(\mathbf{D}_\mathbf{C} - \mathbf{C})$ have been mainly developed by the French school of statisticians (Benali and Escofier 1990,

Chessel and Sabatier 1993, Le Foll 1982, Méot *et al.* 1993) and by Italians in the context of multiscale analysis (Di Bella and Jona-Lasinio 1996). Although these methods have the advantage to produce only positive eigenvalues, they have a major drawback in their objectives: they maximize the local variance (i.e. difference between neighbors) while often users want to minimize this quantity and maximize the spatial correlation.

Wartenberg (1985) was the first to develop a multivariate analysis based on Moran's *I* and autocorrelation. In this article, he proposed to diagonalize $\mathbf{M} = \mathbf{X}^t\mathbf{C}\mathbf{X} / \mathbf{1}^t\mathbf{C}\mathbf{1} = \mathbf{X}^t\mathbf{F}\mathbf{X}$, where $\mathbf{X}$ contains normed and centered variables, and presented examples where the connectivity matrix was based on inter-points distances and always symmetric. Lee (2001) criticizes Wartenberg's work and proves that this approach is not valuable with row-sum standardized weights. He shows that using $\mathbf{W}$, $\mathbf{M}_{jk} \cong \sqrt{SSS_{\mathbf{x}_k}}\, r_{\tilde{\mathbf{x}}_k \mathbf{x}_j}$ and considers rightly that this is not a correct spatial bivariate association measure because it is asymmetric. The derivation of Lee is correct but rather naïve because this formulation yields to an asymmetric matrix $\mathbf{M}$. Finding eigenvalues of such matrix is rather difficult because they can be complex. The correct derivation must use $(\mathbf{W} + \mathbf{W}^t)$ instead $\mathbf{W}$ (de Jong *et al.* 1984). In this case, Wartenberg's approach using $\mathbf{W}$ is exactly the MULTISPATI-(normed) PCA. Equation 13 shows that in this case, the spatial bivariate association measure is symmetric and satisfies the Lee's conditions cited above. Moreover, it is interesting to note that this measure consider the correlations between one original variable and another one lagged variable ($r_{\tilde{\mathbf{y}}_j \mathbf{y}_k}$, $r_{\tilde{\mathbf{y}}_k \mathbf{y}_j}$) and thus is intimately linked to the multivariate extension of Moran scatterplot which plots $\tilde{\mathbf{y}}_j$ versus $\mathbf{y}_k$ or $\tilde{\mathbf{y}}_k$ versus $\mathbf{y}_j$ (Anselin *et al.* 2002).

The MULTISPATI approach maximizes the compromise between the original analysis and the autocorrelation (12). Note that if the constraint $\left\|\mathbf{X}\mathbf{Q}\mathbf{u}_1\right\|_{\mathbf{D}}^2 = 1$ is introduced, the MULTISAPTI analysis will maximize only the spatial part. One alternative for this purpose is to use the principal axes of $\mathbf{X}$ instead of the table $\mathbf{X}$. This corresponds to an implicit use of the Mahalanobis metrics $\mathbf{Q} = (\mathbf{X}^t\mathbf{D}\mathbf{X})^-$ and therefore to a previous decorrelation of the original variables. However, this step requires many observations compared to the number of variables in order to avoid problems of numerical instability. This special case of MULTISPATI on quantitative data is equivalent to the spatial factor analysis and their extensions (Bailey and Krzanowski 2000, Green *et al.* 1988, Grunsky and Agterberg 1991, Switzer and Green 1984).

The two points of view (Geary's *c* and Moran's *I*) have been reconciled by Thioulouse *et al.* (1995) which use the spatial weights from $\mathbf{B}$ to normalize the data. Although this

approach is very elegant in mathematical point of view, it requires that the mean and the variance of the original variables are computed taking into account the spatial connectivity. As summing the elements of **B** by row will not lead usually to uniform weights, the mean and the variance of the original variables are not invariant when permuting the rows of **X**. This is not conceivable because in the case of the null hypothesis (no spatial structure) of inferential tests, it would be required that all spatial units have the same weight in the computation of the mean and variance. This problem does not appear in MULTISPATI test but some works are needed in order to examine the inferential properties of the test.

The MULTISPATI approach is very flexible and can handle various kinds of data. As shown before, many existing methods are particular cases of this approach, which introduce the spatial weight matrix in the statistical triplet notation. This work could be a good starting point to introduce spatial constraint in methods for relating two data tables as canonical correlation analysis (Hotelling 1936), redundancy analysis (Rao 1964) and co-inertia analysis (Dolédec and Chessel 1994, Dray *et al.* 2003a) can also be write using the statistical triplet notation.

Literature cited

Anselin, L. (1995) "Local indicators of spatial association". *Geographical Analysis* 27, 93-115.

Anselin, L. (1996) The Moran scatterplot as an ESDA tool to assess local instability in spatial association. In Fischer, M. M., H. J. Scholten and D. Unwin (eds) *Spatial analytical perspectives on GIS*. London: Taylor and Francis, 111-125.

Anselin, L., I. Syabri and O. Smirnov (2002) "Visualizing Multivariate Spatial Correlation with Dynamically Linked Windows". *New Tools for Spatial Data Analysis: Proceedings of a Workshop*

Bailey, T. C. and W. J. Krzanowski (2000) "Extensions to spatial factor methods with an illustration geochemistry". *Mathematical Geology* 32, 657-682.

Bavaud, F. (1998) "Models for spatial weights: A systematic look". *Geographical Analysis* 30, 153-171.

Benali, H. and B. Escofier (1990) "Analyse factorielle lissée et analyse factorielle des différences locales". *Revue de Statistique Appliquée* 38, 55-76.

Cailliez, F. and J. P. Pagès (1976) *Introduction à l'analyse des données*. Paris: SMASH.

Chessel, D. and R. Sabatier (1993) Couplage de triplets statistiques et graphes de voisinage. In Asselain, B., M. Boniface, C. Duby, C. Lopez, J. P. Masson and J. Tranchefort (eds) *Biométrie et Données spatio-temporelles*. Rennes: Société Française de Biométrie, ENSAR, 28-37.

Cliff, A. D. and J. K. Ord (1973) *Spatial autocorrelation*. London: Pion.

de Jong, P., C. Sprenger and F. van Veen (1984) "On extreme values of Moran's I and Geary's c". *Geographical Analysis* 16, 17-24.

Di Bella, G. and G. Jona-Lasinio (1996) "Including spatial contiguity information in the analysis of multispecific patterns". *Environmental and Ecological Statistics* 3, 269-280.

Dolédec, S. and D. Chessel (1994) "Co-inertia analysis: an alternative method for studying species-environment relationships". *Freshwater Biology* 31, 277-294.

Dray, S., D. Chessel and J. Thioulouse (2003a) "Co-inertia analysis and the linking of ecological data tables". *Ecology* 84, 3078-3089.

Dray, S., D. Chessel and J. Thioulouse (2003b) "Procrustean co-inertia analysis for the linking of multivariate data sets". *Ecoscience* 10, 110-119.

Dray, S., N. Pettorelli and D. Chessel (2003c) "Multivariate analysis of incomplete mapped data". *Transactions in GIS* 7, 411-422.

Escoufier, Y. (1987) The duality diagramm : a means of better practical applications. In Legendre, P. and L. Legendre (eds) *Developments in numerical ecology*. Berlin: Springer Verlag, 139-156.

Geary, R. C. (1954) "The contiguity ratio and statistical mapping". *The incorporated Statistician* 5, 115-145.

Getis, A. and D. A. Griffith (2002) "Comparative spatial filtering in regression analysis". *Geographical Analysis* 34, 130-140.

Goodall, D. W. (1954) "Objective methods for the classification of vegetation III. An essay on the use of factor analysis". *Australian Journal of Botany* 2, 304-324.

Green, A. A., M. Berman, P. Switzer and M. D. Graig (1988) "A transformation for ordering multispectral data in terms of image quality with implications for noise removal". *IEEE Transactions on Geoscience and Remote Sensing* 26, 65-74.

Greenacre, M. J. (1984) *Theory and Applications of Correspondence Analysis*. London: Academic Press.

Griffith, D. A. (1996) "Spatial autocorrelation and eigenfunctions of the geographic weights matrix accompanying geo-referenced data". *Canadian Geographer* 40, 351-367.

Griffith, D. A. (2000a) "Eigenfunction properties and approximations of selected incidence matrices employed in spatial analyses". *Linear Algebra and its Applications* 321, 95-112.

Griffith, D. A. (2000b) "A linear regression solution to the spatial autocorrelation problem". *Journal of Geographical Systems* 2, 141-156.

Grunsky, E. C. and F. P. Agterberg (1991) "SPFAC: a FORTRAN-77 program for spatial factor analysis of multivariate data". *Computers & Geosciences* 17, 133-160.

Hotelling, H. (1933) "Analysis of a complex of statistical variables into principal components". *Journal of Educational Psychology* 24, 417-441.

Hotelling, H. (1936) "Relations between two sets of variates". *Biometrika* 28, 321-377.

Ihaka, R. and R. Gentleman (1996) "R: A Language for Data Analysis and Graphics". *Journal of Computational and Graphical Statistics* 5, 299-314.

Kadmon, R. and A. Danin (1997) "Floristic variation in Israel: a GIS analysis". *Flora* 192, 341-345.

Kazi-Aoual, F., S. Hitier, R. Sabatier and J. D. Lebreton (1995) "Refined approximations to permutation tests for multivariate inference". *Computational Statistics and Data Analysis* 20, 643-656.

Kiers, H. A. L. (1994) "Simple structure in component analysis techniques for mixtures of qualitative and quantitative variables". *Psychometrika* 56, 197-212.

Krishna Iyer, P. V. (1949) "The first and second moments of some probability distributions arising from points on a lattice and their application". *Biometrika* 36, 135-141.

Le Foll, Y. (1982) "Pondération des distances en analyse factorielle". *Statistique et Analyse des données* 7, 13-31.

Lebart, L. (1969) "Analyse statistique de la contiguïté". *Publication de l'Institut de Statistiques de l'Université de Paris* 28, 81-112.

Lee, S.-I. (2001) "Developing a bivariate spatial association measure: An integration of Pearson's r and Moran's I". *Journal of Geographical Systems* 3, 369-385.

Méot, A., D. Chessel and R. Sabatier (1993) Opérateurs de voisinage et analyse des données spatio-temporelles. In Lebreton, J. D. and B. Asselain (eds) *Biométrie et environnement*. Paris: Masson, 45-72.

Moran, P. A. P. (1948) "The interpretation of statistical maps". *Journal of the Royal Statistical Society Series B-Methodological* 10, 243-251.

Ord, J. K. (1975) "Estimation methods for models of spatial interaction". *Journal of the American Statistical Association* 70, 120-126.

Rao, C. R. (1964) "The use and interpretation of principal component analysis in applied research". *Sankhya A* 26, 329-359.

Switzer, P. and A. A. Green (1984) "Min/max autocorrelation factors for multivariate spatial imagery". Technical report 6, Stanford University, Stanford.

Tenenhaus, M. and F. W. Young (1985) "An analysis and synthesis of multiple correspondence analysis, optimal scaling, dual scaling, homogeneity analysis and other methods for quantifying categorical multivariate data". *Psychometrika* 50, 91-119.

Thioulouse, J., D. Chessel and S. Champely (1995) "Multivariate analysis of spatial patterns: a unified approach to local and global structures". *Environmental and Ecological Statistics* 2, 1-14.

Tiefelsdorf, M., D. A. Griffith and B. Boots (1999) "A variance-stabilizing coding scheme for spatial link matrices". *Environment and Planning A* 31, 165-180.

Torre, F. and D. Chessel (1995) "Co-structure de deux tableaux totalement appariés". *Revue de Statistique Appliquée* 43, 109-121.

Wartenberg, D. (1985) "Multivariate spatial correlation: a method for exploratory geographical analysis". *Geographical Analysis* 17, 263-283.

Table 1

Eigenvalues, variance, values and Moran's *I* computed for the first four axes of the classical multivariate analysis (MIX) and for the spatially constrained analysis. Probability of the test of significance of Moran's *I* (9999 permutations) are also given.

	λ	variance	***I***	*p*
MIX1	4.6223	4.6223	0.5106	0.0001
MIX2	1.9026	1.9026	0.5009	0.0002
MIX3	1.4263	1.4263	0.2532	0.0262
MIX4	1.0478	1.0478	0.1217	0.1385
MS1	2.5562	4.3285	0.5905	0.0001
MS2	1.1309	1.7765	0.6366	0.0001
MS3	0.3687	1.2018	0.3068	0.0095
MS4	0.2113	0.7699	0.2745	0.0208

Figures

Figure 1

Analysis of Irish county data. Bar plot of eigenvalues for (a) PCAMIX and (b) MULTISPATI-PCAMIX.

Figure 2

Analysis of Irish county data. Projections the first four principal axes of PCAMIX onto the first two principal axes of MULTISPATI-PCAMIX. As principal axes

are normed, this figure represents correlations.

Figure 3

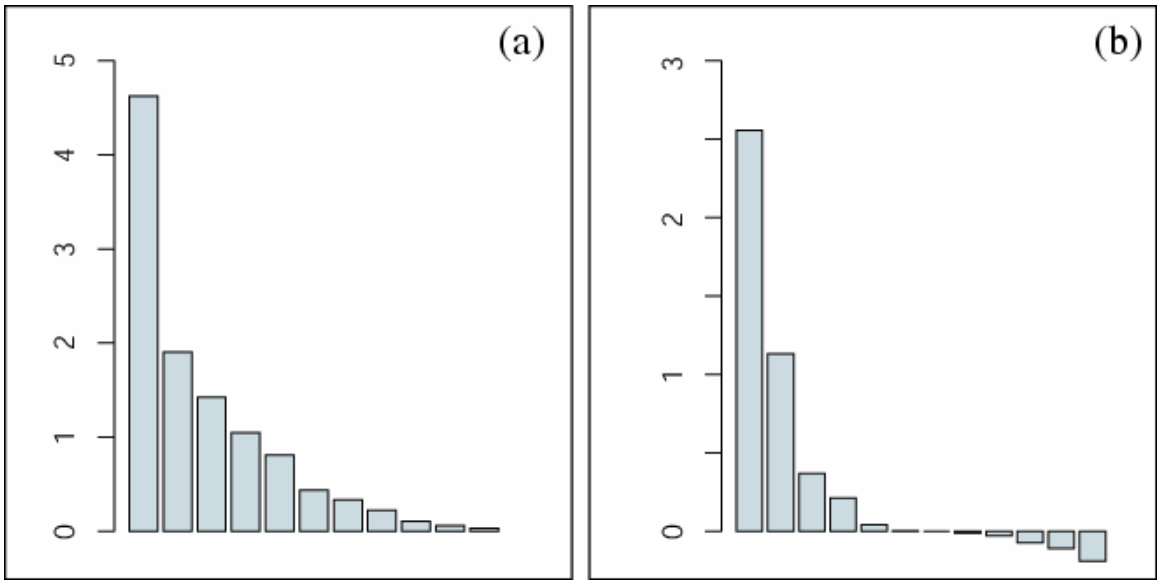
Analysis of Irish county data. Scores of variables for the first two axes of (a) PCAMIX and (b) MULTISPATI-PCAMIX. Codes of variables are given (in bold): valuation level (**tax:** high (**hi**), medium (**me**), low (**lo**)) based on percentage number of agricultural holdings in valuation groups (see text for explanations); number of milch **cow**s, **pig**s, **sheep** and **other** cattle per 1 000 acres crops and pasture; town and village population as percentage of total (**town.pop**); number of private **car**s registered and number of **radio** licences per 1 000 population; single males as % of all males aged 30-34 (**single.man**); retail **sales** in £ per person. The values of d indicates the size of squares of the grid.

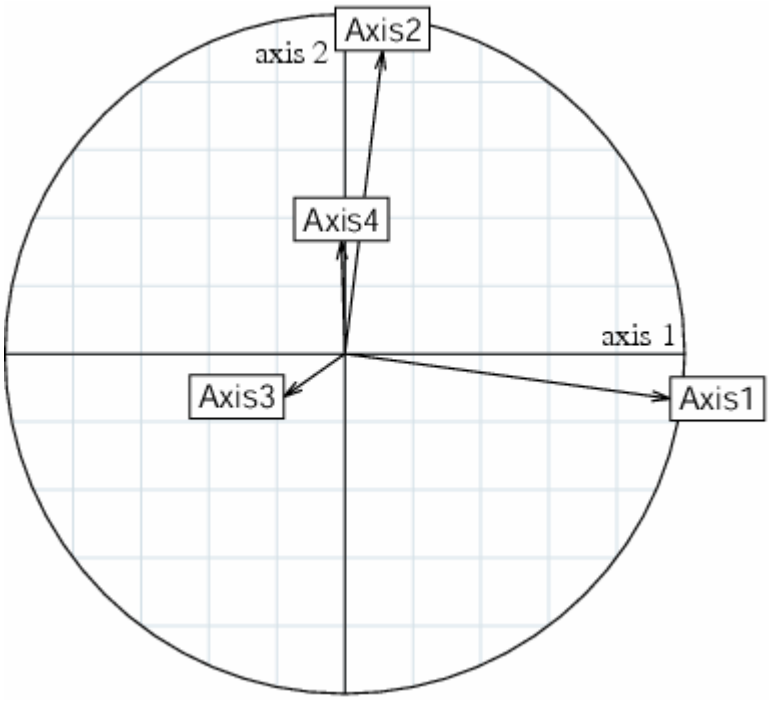
Figure 4

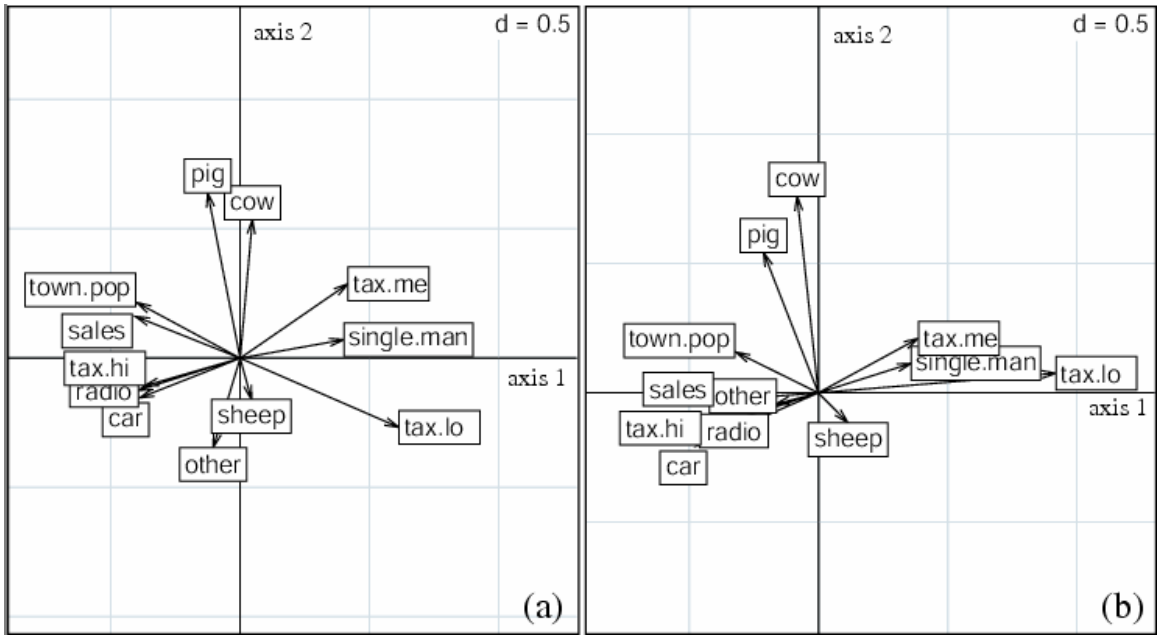
Analysis of Irish county data. Scores and lagged scores of counties for the first two axes of (a) PCAMIX and (b) MULTISPATI-PCAMIX. For each county, the arrow links the score to the lagged score (averages of neighbors weighted by the spatial connection matrix). A partition into three groups (c) has been obtained by hierarchical clustering (complete linkage, Euclidean distance) on the first two scores and lagged scores of MULTISPATI-PCAMIX. Code for counties are the same as those used by Geary (1954). The values of d indicates the size of squares of the grid.

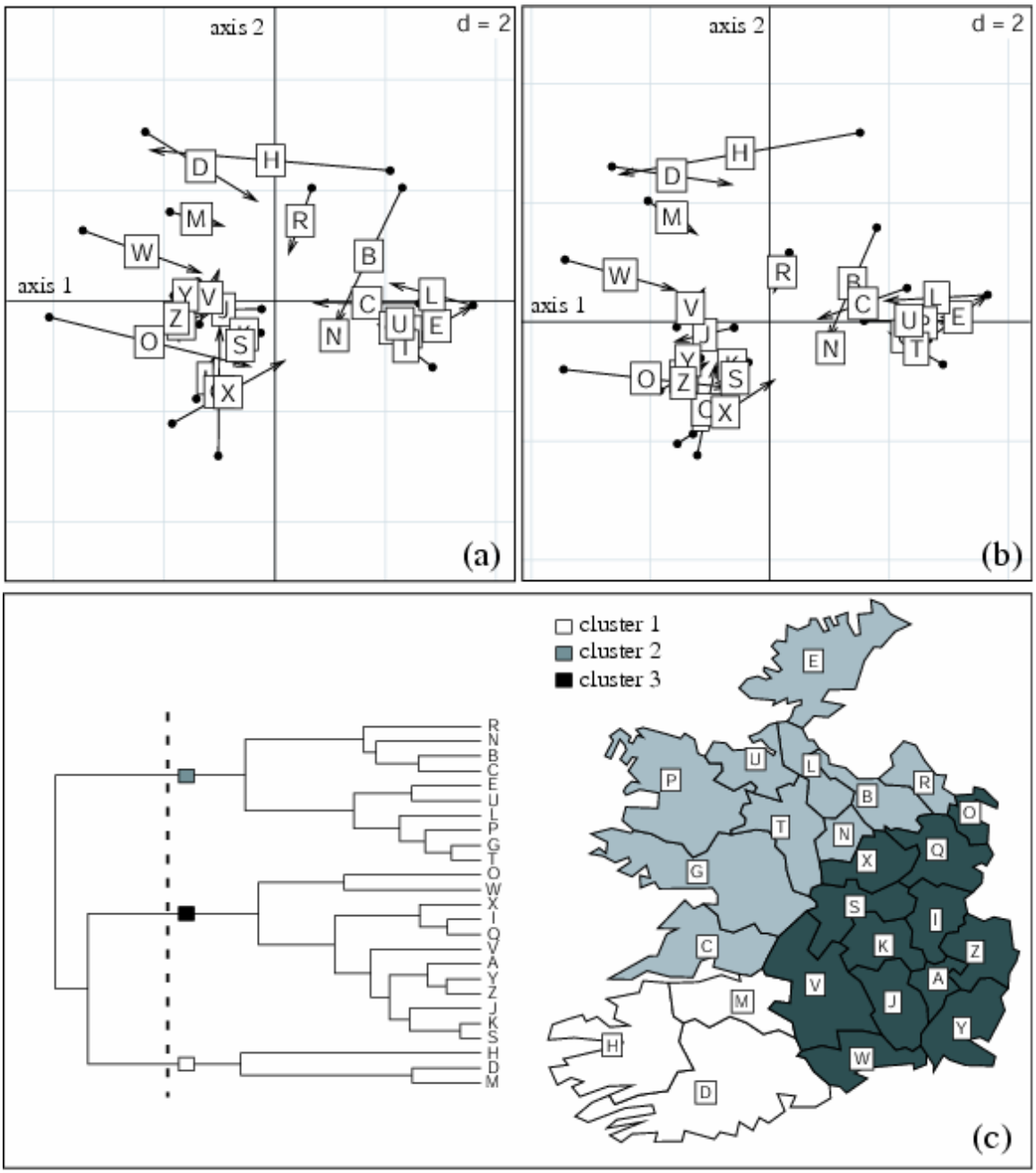
Figure 5

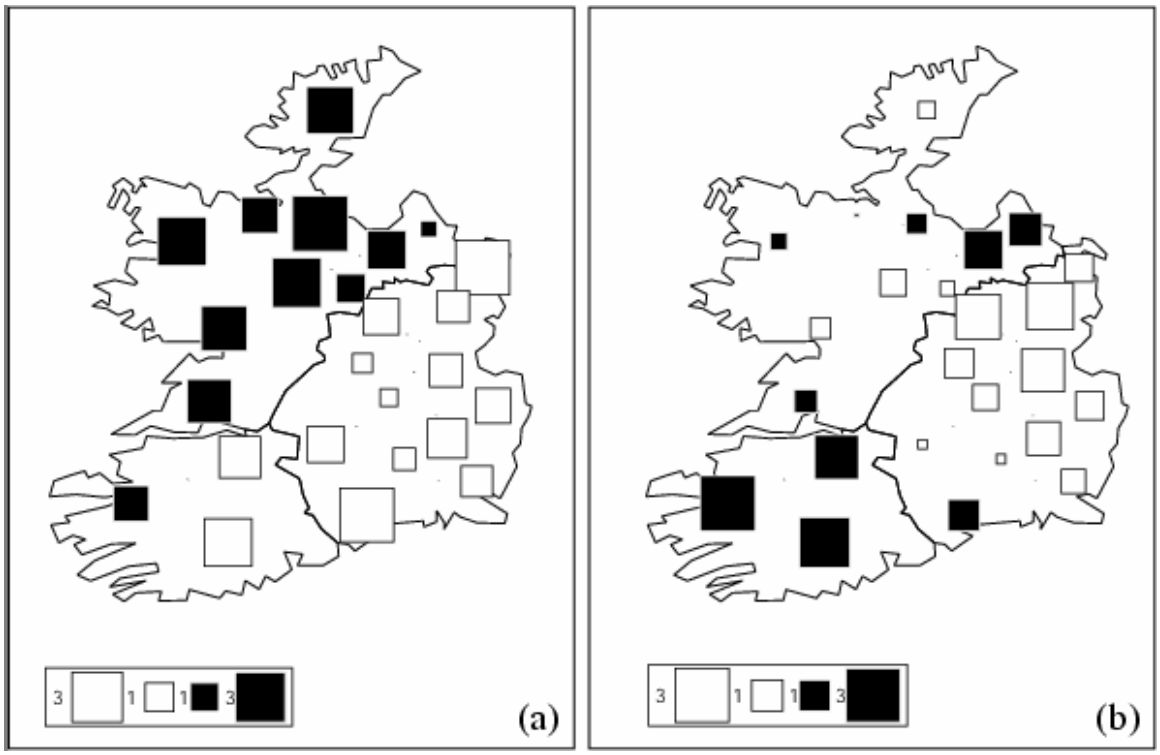
Analysis of Irish county data. Mapping of scores of counties for (a) the first and (b) the second axis of MULTISPATI-PCAMIX.


(a)
0
1
2
3
4
5
(b)
0
1
2
3


Axis2
axis 2
Axis4
axis 1
Axis3
Axis1


axis 2
d = 0.5
pig
cow
town.pop
tax.me
sales
single.man
tax.hi
radio
car
sheep
tax.lo
other
axis 1
(a)
axis 2
d = 0.5
cow
pig
town.pop
tax.me
single.man
tax.lo
sales
other
tax.hi
radio
sheep
car
axis 1
(b)


axis 2
d = 2
axis 1
(a)
(b)
cluster 1
cluster 2
cluster 3
(c)


3
1
1
3
(a)
3
1
1
3
(b)

2. A generalized, variogram-based framework for multi-scale ordination

auteurs

Pierre Couteron
Sébastien Ollier

résumé

Multi-scale ordination (MSO) deals with potential scale-dependence in species assemblages, by studying how results from multivariate ordination may be different on different spatial scales. MSO methods were initially based on two-term local covariances between species, and therefore required sampling designs composed of adjacent quadrats. A variogram-based MSO, applicable to very diverse sampling designs has recently been introduced by H.H. Wagner (2003, 2004). This refers to Principal Component Analysis, Correspondence Analysis and derived "two-table" (also called "direct") ordination methods, i.e., Redundancy Analysis and Canonical Correspondence Analysis.

In this paper we put forward an enlarged framework for variogram-based MSO which relies on a generalized definition of inter-species covariance and on matrix expression of spatial contiguity between sampling units. This enables us to provide distance-explicit decompositions of variances and covariances (in their generalized meaning), that are consistent with many ordination methods in both their single- and two-table versions. A spatially explicit apportioning of diversity indices is proposed for some particular definitions of variance. Referring to two-table ordination methods allowed the multi-scale study of residual spatial patterns after factoring out of available environmental variables. Some aspects of the approach are briefly illustrated with vegetation data from a neotropical rainforest in French Guiana.

mots-clés

diversity apportioning, species assemblages, multi-scale ordination, multivariate geostatistics, canonical correspondence analysis, spatial contiguity, tropical rainforest, variogram

journal

Ecology: accepté, sous presse

Introduction

Determining to what extent multi-species patterns of association may be different on different spatial scales is obviously a central issue in ecology (Levin1992). The concern to integrate space in numerical studies of inter-species association has led to the development of a method of multi-scale ordination (MSO; Ver Hoef and Glenn-Lewin 1989). This requires data from continuous sampling designs (e.g. belt transects) since it is based on the computation of two- term local covariances between species (Greig-Smith 1983). In two recent papers, H.H. Wagner (2003, 2004) proposed a new method for MSO based on the variogram (Wackernagel 1998) thereby allowing the use of data collected by means of very diverse sampling designs. This insightful approach was proposed for two usual ordination methods, i.e., Principal Component Analysis (PCA) and Correspondence Analysis (CA). Extension to "direct" ordination methods (Legendre and Legendre 1998), using two data tables, such as Redundancy Analysis (RDA, relating to PCA) and Canonical Correspondence Analysis (CCA, relating to CA) was also proposed by Wagner (2004).

This most recent contribution is a considerable step forward since CA is generally preferred to PCA for the study of inter-species associations. CA is a very popular and powerful method that positions species and sites along common ordination axes, by applying the same centering and weighting options to rows and columns of the site by species table. However, in spite of several attractive properties, there is no reason to consider CA as being automatically the most appropriate ordination method whatever the characteristics of the data and the aim of the study (Gimaret-Carpentier et al. 1998). Several alternatives to CA, with distinct properties, can be defined by changing weighting options for either sites or species. For instance, Pélissier et al. (2003) demonstrated that changing species weighting, i.e. placing varying degrees of emphasis on scarce species, could be used to define three ordination methods (including CA) and that each was consistent with one classical diversity index (richness, Shannon's, Simpson's). On the other hand, Dolédec et al. (2000) used a uniform weighting of sites to derive an alternative to CCA with interesting properties for the separation of species niches. Hence, it would be preferable for ecologists to become aware of the potential adaptability of both single- and two-table methods of ordination to their specific aims and to the characteristics of their data. Such adaptability should also encompass the emerging field of spatially explicit ordinations.

The paper presented here was triggered by the pioneering work conducted by Wagner (2003, 2004) but aims to define a broader framework for variogram-based multi-scale

ordinations. We intend to demonstrate that it is possible to partition by distance the results of very diverse ordination methods, as defined by re-scaling and weighting options for the rows and columns of the data tables. To do so, we will introduce a generalized definition of covariance between species which encompasses several ordination methods while being amenable to scale-explicit decompositions. Consistency with the additive decomposition of common diversity indices (Pélissier et al. 2003; Couteron and Pélissier in press) will be highlighted, while referring to methods of two-table "direct" ordination will allows the explicit analysis of residual spatial patterns after the factoring out of some environmental variables. This aspect will be emphasized in a brief illustration based on vegetation data from a neotropical rainforest. In this report, we have chosen to keep mathematical developments to a minimum while providing a complete treatment in matrix form in an appendix submitted to the Electronic Ecological Archives. Computer programs are freely available from the first author (Matlab® version) or on http://pbil.univ-lyon1.fr/CRAN/ (R [Ihaka and Gentleman 1996] version integrated in the ade4 package).

A generalized definition of covariance

Data tables containing counts of individual organisms by sampling sites (say "quadrats") and taxa (usually species) are both a central and general feature of ecological studies. Let us consider such a table, based on N sampled individuals, for which f_{ai} is the total number of individuals belonging to species i ($1<i<S$) that were counted in quadrat a ($1<a<Q$). Let p_{ai} be the corresponding relative frequency ($p_{ai}=f_{ai}/N$) while p_{a+} and p_{+i} are the relative frequencies for quadrat a and species i, respectively. We have introduced our topic with explicit reference to counted individuals, though the above parameters remain meaningful as long as f_{ai} is a non-negative value (biomass measurements, semi-quantitative indices of abundance, presence/absence, …) expressing the abundance of species i in quadrat.

Ordination methods such as correspondence analysis CA and various versions of PCA (ter Braak 1983) are the usual tools employed to analyze quadrats by species tables. Central to all these methods is the application of singular values decomposition (svd), also called eigenanalysis, to a square S by S matrix, which is the usual variance-covariance matrix, **C**, for the species-centered (non-standardized) PCA and which is another matrix $\mathbf{Q}^2$ in the case of CA (see Legendre and Legendre 1998:453 and Wagner 2004 for details). In $\mathbf{Q}^2$, terms on the diagonal are homologous to variances and are proportional to the portions of the total chi-square of the data table (Legendre and Legendre 1998:452) that are attached to each of the S

species. Off-diagonal terms are homologous to the usual pairwise covariances and measure to what extent two arbitrary species may conjointly depart from expected abundance values.

We can see matrices **C** and **Q²** as nothing more that special cases of a square matrix $\mathbf{G_T} = \left[g_{ij} \right]_{1 \le i \le S,\, 1 \le j \le S}$ based on an appropriate generalization of the notions of species variance (diagonal values) and covariance (off-diagonal values). This generalized measure of covariance is, for two arbitrary species *i* and *j*:

$$g_{ij} = \frac{1}{2} \sum_{a=1}^{Q} \sum_{b=1}^{Q} (x_{ai} - x_{bi}) \sqrt{w_i w_j} (x_{aj} - x_{bj}) \delta_a \delta_b \qquad (1)$$

variance being a special case with $i=j$.

Here, w_i weighs the influence of species *i*, while δ_a and δ_b are the weights given to quadrats *a* and *b*, respectively; x_{ai} denotes any measure of abundance of species *i* in quadrat *a* that can be derived from the initial value f_{ai} via re-scaling options (Table 1).

The choice of weighting options is a central yet often eluded question when using multivariate techniques, since weighting along with re-scaling and centering defines the nature of the distance between quadrats and, for some methods, also between species. Moreover, the choice of weighting options relates to very practical questions concerning, for instance, the influence that it seems meaningful to confer to a particular species in the definition of a multi-specific assemblage, or to a given quadrat in the investigation of an ecological gradient. Addressing such questions means that the biogeographic context must be taken into account (e.g., are there many scarce species? How abundant are the most frequent species?) along with the sampling design (does it give a fair estimate of species abundance in a region?) and, for two-table methods, the nature of the ecological gradients under study (are there strong limiting factors or threshold effects?). More detailed discussions on the consequences of weighting can be found in Dolédec et al. (2000, regarding quadrats in direct gradient analysis) and in Pélissier et al. (2003, regarding species).

Combining re-scaling and weighting options opens up a wide selection of ordination methods and associated properties. Some examples, based on published methods, are presented in Table 1, but other possibilities are obviously imaginable. The presentation of classical ordination methods in terms of weighting of rows and columns was introduced by Escoufier (1987) and was used by Sabatier et al. (1989) and Dolédec et al. (2000) for several single- and two-table methods (including CA, CCA and classical versions of PCA and RDA). Pélissier et al. (2003) used this presentation to compare the properties of the three methods corresponding to option IV in Table 1, and to investigate their relationship with diversity

measures. All these authors based their presentation of the methods on a species-centered version of the data table containing differences between individual observations, x_{ai}, and the δ_a-weighted mean value, $\bar{x}_i$, found for each species. Alternatively, in Eq.1, we use all pairwise differences between observations to compute variance and covariance. The equivalence of the two approaches is explained in the appendix (see Eq. A5 to A10).

Generalized spatial covariance

From Eq.1, the contribution made by a given pair (*a,b*) of quadrats to the covariance between two species can be expressed as:

$$g_{ij}(a,b) = \frac{1}{2}(x_{ai} - x_{bi})\sqrt{w_i w_j}(x_{aj} - x_{bj})\delta_a\delta_b \qquad (2)$$

This translates easily into a generalized version of cross-variograms ($i \neq j$) and variograms ($i=j$), namely:

$$\gamma_{Gij}(h) = \frac{1}{K(h)} \sum_{a,b|h_{ab}\approx h} g_{ij}(a,b) \qquad (3)$$

where h is the central value of a given distance class, and where $K(h)$ is a scaling coefficient, such as: $K(h) = \sum_{a,b|h_{ab}\approx h} \delta_a\delta_b$

(4)

Considering all species together leads to a generalized variogram of species composition ("generalized" since potentially relating to several ordination methods and distance metrics):

$$\gamma_G(h) = \sum_i \gamma_{Gii}(h) \qquad (5)$$

Eq.4 is a crucial point since $K(h)$ standardizes $\gamma_G(h)$ in such a manner to equate its expected value (sill) with the total variance of the ordination method defined by weighting options. This is completely different from computing the usual experimental variogram from ordination scores, except for the special case of uniform quadrat weights (as in Wagner 2003) where $K(h)$ is proportional to the number, n_h, of pairs of quadrats relating to distance class h. Conversely, if quadrat weights are not uniform, scaling by $K(h)$ is the only manner to ensure that, whatever the distance class, the expected value of $\gamma_G(h)$ is the total variance ("inertia") attached to matrix $\mathbf{G_T}$ and computed from the sum of its diagonal elements. Note that such a

property is not guaranteed by the manner in which Wagner (2004) defined her version (denoted as $\gamma_Q(h)$) of the CA-related variogram since the corresponding scaling remains proportional to n_h despite the fact that quadrat weights are not uniform. The scaling by *K(h)* is of particular interest if weightings of both species and quadrats are chosen as to relate to a diversity measurement (option IV in Table 1). In this case, the trace of $\mathbf{G_T}$ is the diversity among quadrats (Couteron and Pélissier, in press), which means that $\gamma_G(h)$ measures the average beta-diversity between pairs of quadrats corresponding to distance class *h*. Equivalently,

$$VAR(a,b) = \sum_i g_{ii}(a,b) \qquad (6)$$

quantifies the contribution made by a given couple *(a,b)* of quadrats to beta-diversity. Some classical dissimilarity indices, such as Jaccard's or Sorensen's (Legendre and Legendre 1998:256) are often used to quantify beta-diversity, though these have no direct connection with either geostatiscal tools or ordination methods. Conversely, Eqs.2 and 6 provide a family of dissimilarity indices some of which relate directly to both.

Variograms and cross-variograms of ordination axes

Regardless of the reference ordination method chosen, a generalized variance-covariance matrix, $\mathbf{G}_h$, is computed for each distance class *h*. To ensure efficient computations by any matrix-oriented programming language, as we did with Matlab® and R (Ihaka and Gentleman 1996), we introduced a matrix formulation of the method. It is based of a contiguity relationship (Thioulouse et al. 1995) consistent with the variogram, which considers two quadrats as "neighbors at scale *h*" if the distance between them is within the bounds of the class centered around *h* (see Appendix). Assuming that distance classes include all pairs of quadrats while being mutually exclusive, we demonstrated (see Appendix, Eq. A13) that the matrices $\mathbf{G}_h$ sum to $\mathbf{G_T}$, whatever the initial choice of the reference ordination method by weighting options.

The eigenvectors and eigenvalues originating from the singular values decomposition (svd) of $\mathbf{G_T}$ can be partitioned with respect to distance classes (Appendix), as a generalization of the fundamental principle introduced by Ver Hoef and Glenn-Lewin (1989). But the complete variance-covariance matrix, $\mathbf{F}_h$, between eigenvectors can also be obtained (Appendix, Eq.A13). Considering off-diagonal elements of $\mathbf{F}_h$, namely covariances at scale *h* between eigenvectors, is a new perspective in MSO which can be used to investigate the

potential existence of a scale-dependent covariance between distinct ordination axes. This question, though ignored by most papers devoted to MSO, is closely related to the initial concern of Noy-Meir and Anderson (1971), namely that ordination results may substantially vary with spatial scales. This would mean, for example, that species displaying the most prominent variations of abundance may not be the same depending on the average distance between the quadrats, or that distinct species assemblages may be found for different distance classes. How can we test whether this is the case or not? One way would be to carry out an ordination for each of the $\mathbf{G}_h$ matrices and compare the results, but this is likely to be cumbersome while objective criteria for the comparison are not straightforward to define. We propose a more efficient approach by constructing cross-variograms of eigenvectors from the off-diagonal values of $\mathbf{F}_h$ matrices after appropriate scaling by $K(h)$. All these cross-variograms have an expectation of zero, since the eigenvectors of $\mathbf{G_T}$ are globally uncorrelated, but some may have significant departures from this expectation on particular scales. (Of course, only the cross-variograms for the most prominent eigenvectors are to be analyzed.) If this is the case, it is possible to know at which scales it may be worthwhile carrying out specific ordination analyses via the svd of the corresponding $\mathbf{G}_h$ matrices.

Taking environmental heterogeneity into account

If the species by quadrats table is accompanied by environmental variables assessed at the quadrat scale, it is advantageous to factor out the influence of such variables prior to analyzing the residual spatial patterns of species composition. Technically, this verifies whether some basic assumptions, such as "intrinsic" stationarity (used to interpret the empirical variogram), or independence of residuals (assumed to fit a linear model of species-environment relationship) are met by the data (see Wagner 2004 for an extensive discussion). In terms of ecological interpretation, it is judicious to see residual spatial patterns of community composition as predominantly shaped by biotic processes, such as species dissemination or species interactions (Wagner 2004) and as potentially informative on the scale at which such processes may operate.

Any two-table "direct" ordination starts from the decomposition of the quadrats by species table, **X**, into an approximated table **A**, modelled from environmental variables by a weighted linear regression and a residual table **R**. Such a decomposition may be carried out in a manner consistent with a two-table version of any of the ordination methods mentioned in Table 1 (Sabatier et al. 1989; Pélissier et al. 2003) when the linear regression uses the quadrat

weights defining the ordination. (For instance, defining table **X** along with species and quadrat weights so as to make them consistent with CA means that an ordination on **A** would be a CCA.) To study residual spatial patterns, it is possible to break down $\mathbf{G_R}$, i.e., the variance-covariance matrix computed from **R,** into additive variance-covariance matrices, $\mathbf{G_{R}}_h$, each corresponding to a certain distance class, and on which a variogram-based multi-scale analysis can be based (see Appendix).

Brief illustration based on tropical rain forest data

We considered 7,189 trees (diameter at breast height above 10 cm) belonging to 59 species sampled in a lowland tropical rain forest of ca. 10, 000 ha in French Guiana. The sampling design was based on 411 rectangular quadrats of 0.3 ha each, located at the nodes of a 400 m by 500 m grid. Environmental information at the quadrat scale was expressed by a synthetic nominal variable (12 categories) primarily based on topography and soil water regime (see Couteron et al. 2003 for details). Performing CA on the quadrats by species table showed two main floristic gradients corresponding to the second and third axes (CA2 and CA3). The first axis (CA1) resulted from the spurious occurrence of a scarce species (17 trees) in a particular quadrat and this illustrates a well-known drawback of CA. Results of the non-symmetric correspondence analysis (NSCA) were free from this problem, while the two main axes, NSCA1 and NSCA2, correlated strongly with CA3 and CA2 ($r = 0.76$ and $r = 0.84$, respectively) despite being defined from distinct species. For this data set, shifting emphasis from scarce to abundant species changed the hierarchy between the ordination axes, but the detection of two main floristic gradients proved robust with respect to species weighting. To go beyond these results established by a previous study (Couteron et al. 2003) we explicitly considered inter-quadrats distances by applying the generalized variogram-based MSO with CA and NSCA as reference ordination methods. First, we partitioned the total variance attached to each ordination axis (eigenvalue) among distance classes. Since diversity-related ordinations were used (Pélissier et al. 2003), it was the total among-quadrats diversity (*sensu* the species richness for CA or the Simpson-Gini index for NSCA) which was successively broken down with respect to main floristic gradients (eigenvalues) and distance classes.

The floristic gradient defined by CA2 and NSCA2 failed to show any obvious spatial pattern since variograms were found to waver between confidence envelopes, and this regardless of the reference ordination (Fig.1-a and b). The study of residual patterns, after

factoring out the 12 environmental categories (unconstrained ordinations on the residual variance-covariance matrix, **R**) showed significant departures of the CA-based variogram for distances under 2 km. Such a change in the variogram stemming from the partialling out of the environmental variable typifies the complex interaction which can be expected between the environmental heterogeneity and spatial patterns of species assemblages. It also illustrates the advantage of studying such an interaction within a unified theoretical framework of multi-scale ordination since it enabled us to express in the same unit all kinds of results derived from a particular ordination method. This renders variograms of both initial and residual patterns directly comparable, i.e. a desirable property that could not have been achieved by the computation of classical variograms from ordination scores. The other floristic gradient (defined by CA3 and NSCA1) showed a strong spatial pattern that pointed toward non-stationarity (see Wagner 2004 for a detailed definition) since both initial variograms (Fig.1-c and 1-d) continued to rise up to 8 km without reaching a sill. The variograms of the homologous axes provided by CA and NSCA after factoring out the qualitative environmental categories appeared to be very similar. This indicated that the observed spatial patterns relating to this floristic gradient were not determined by the spatial distribution of the environmental categories.

By separately analyzing spatial patterns of distinct ordination axes we have implicitly hypothesized the absence of any scale-dependent relationship between the ordination axes or, equivalently, the stability across scales of inter-species covariances ("intrinsic" covariances *sensu* Wackernagel, 1998). Such a hypothesis can be easily addressed by computing the cross-variograms between the ordination axes (from off-diagonal elements of matrices Fh, Eq.A18 in the Appendix). Only cross-variograms computed from the residual variance-covariance matrix (R) are shown (Fig.1-e and –f) since homologous cross-variograms from the initial data table were very similar. No scale dependence was observed between the two ordination axes given by NSCA (Fig.1-f) and covariances between abundant species thus appeared to be stable across scales. This was not the case when the emphasis was placed on scarcer species by the use of CA since most values of the corresponding cross-variogram were outside the confidence envelopes (Fig.1-e). Indeed, by diagonalizing the pooled variance-covariance matrices for distances under 4 km vs. distances above 4 km, we obtained two clearly distinct sets of species with high loadings on the ordination axes (results not shown). This result exemplified how scale-dependence may be detected by analyzing cross-variograms between ordination axes, while also illustrating the influence that species weighting may have: CA results proved scale-dependent though NSCA results did not.

Concluding remarks

In the above illustration, we deliberately restricted ourselves to some particular analyses that can be obtained from the generalized variogram-based MSO, but other kinds of analyses are clearly possible. For instance, it may be of interest to compare the spatial patterns of all individual species and identify scales on which some patterns may differ from others. This can be done by analysing, for instance by PCA, the table containing the generalized variograms of all species (prior standardization by variances of individual species as to have all sills equal to one is likely to be preferable). It may also be of interest to study the manner in which the distribution of the eigenvalues changes with scale, by analyzing the table containing the generalized variogram of the eigenvalues. Any type of MSO can address these questions, but our unifying approach also provides a choice between ordination methods while offering links with diversity measurement and apportioning. As a consequence, future users may be able to select the particular ordination method (either direct or indirect) that best suits their data and aims.

Furthermore, the presentation of the method in matrix-form, with the use of contiguity matrices (Appendix), not only allows for efficient programming, but also opens up interesting methodological perspectives. Indeed, two-term local variances and covariances (TTLV/TTLC) on which the "classical" MSO relies (Ver Hoef and Glenn-Lewin 1989) have been also formulated using contiguity matrices (Di Bella and Jona-Lasinio 1996, Ollier et al. 2003). Such a formulation is obviously a sound basis, not only for a further generalization of the TTLV/TTLC-based MSO, as performed by us with the variogram-based MSO, but also for a thorough investigation of the respective properties of the two approaches.

Acknowledgements

We are indebted to D. Chessel (University of Lyon-1) for important insights into several topics mentioned in this paper, to R. Pélissier (IRD/UMR AMAP) for his valuable comments on preliminary drafts, and to H. H. Wagner for pertinent comments on the initial version.

Literature cited

Couteron, P., and R. Pélissier. Additive apportioning of species diversity: towards more sophisticated models and analyses. Oikos, in press.

Couteron, P., R. Pélissier , D. Mapaga, J.-F. Molino, and L. Teillier. 2003. Drawing ecological insights from a management-oriented forest inventory in French Guiana. Forest Ecology and Management 172:89-108.

Di Bella, G., and G. Jona-Lasinio. 1996. Including spatial contiguity information in the analysis of multispecific patterns. Environmental and Ecological Statistics **3**: 269-280.

Dolédec, S., D. Chessel, and C. Gimaret-Carpentier. 2000. Niche separation in community analysis: a new method. Ecology **81**:2914–2927.

Escoufier, Y. 1987. The duality diagram: a means of better practical applications. Pages 139-156 *in* P. Legendre and L. Legendre, editors. Development in numerical ecology. Springer-Verlag, Berlin, Germany.

Gimaret-Carpentier, C., D. Chessel, and J.-P. Pascal. 1998. Non-symmetric correspondence analysis: an alternative for species occurrences data. Plant Ecology **138**:97–112.

Greig-Smith, P. 1983. Quantitative Plant Ecology. 3rd ed. Blackwell Science Publishing, Oxford, UK.

Ihaka, R., and R. Gentleman. 1996. R: a language for data analysis and graphics. Journal of Computational and Graphical Statistics **5**:299-314.

Legendre, P., and L. Legendre. 1998. Numerical ecology. Elsevier, Amsterdam, The Netherlands.

Levin, S. 1992. The problem of pattern and scale in ecology. Ecology **73**: 1943-1967.

Noy-Meir, I., and D. Anderson. 1971. Multiple pattern analysis or multiscale ordination: towards a vegetation hologram. Pages 207-232 in G. P. Patil, E.C. Pielou, and E.W. Water, editors. Statistical ecology: populations, ecosystems, and systems analysis. Pennsylvania State University Press, University Park, Pennsylvania, USA.

Ollier, S., D. Chessel, P. Couteron, R. Pélissier, and J. Thioulouse. 2003. Comparing and classifying one-dimensional spatial patterns: an application to laser altimeter profiles. Remote Sensing of Environment **85**: 453:462.

Pélissier, R., P. Couteron, S. Dray, and D. Sabatier. 2003. Consistency between ordination techniques and diversity measurements: two strategies for species occurrence data. Ecology, **84**:242-251.

Sabatier, R., J.-D. Lebreton, and D. Chessel. 1989. Principal component analysis with instrumental variables as a tool for modelling composition data. Pages 341–352 *in* R. Coppi and S. Bolasco, editors. Multiway data analysis, Elsevier Science Publishers, The Netherlands.

ter Braak, C. J. F. 1983. Principal components biplots and alpha and beta diversity. Ecology 64: 454-462.

Thioulouse, J., D. Chessel, and S. Champély. 1995. Multivariate analysis of spatial patterns: a unified approach to local and global structures. Environmental and Ecological Statistics **2**: 1-14.

Ver Hoef, J. M., and D. C. Glenn-Lewin. 1989. Multiscale ordination: a method for detecting pattern at several scales. Vegetatio **82**: 59-67.

Wackernagel, H. 1998. Multivariate geostatistics. Second completely revised edition. Springer, Berlin, Germany.

Wagner, H. H. 2003. Spatial covariance in plant communities: integrating ordination, geostatistics, and variance testing. Ecology **84**: 1045-1057.

Wagner, H. H. 2004. Direct multi-scale ordination with canonical correspondence analysis. Ecology **85**: 342-351.

Table 1

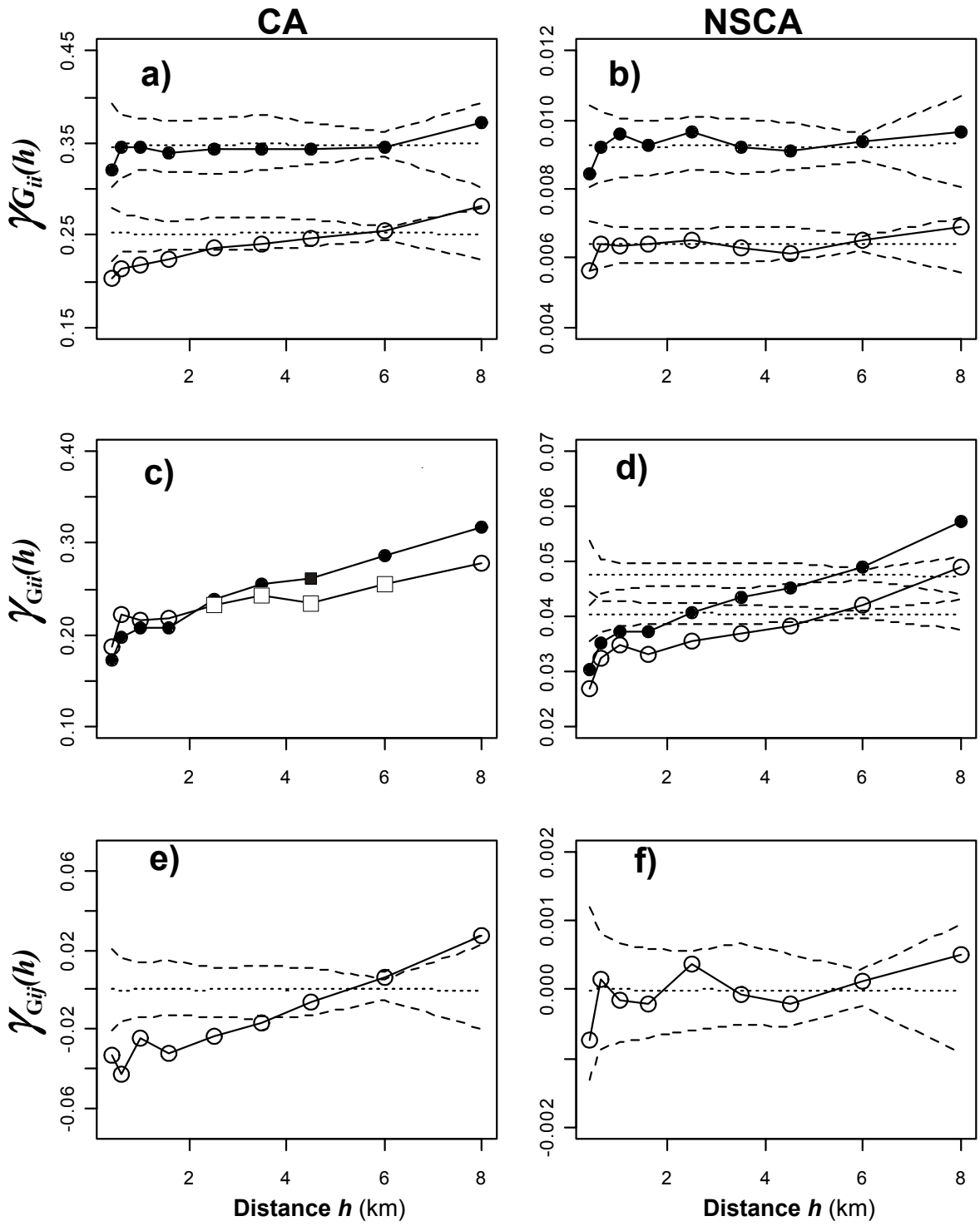
Definition of some ordination methods from re-scaling and weighting options. D is the total diversity computed from the species frequency distribution ($D = \sum_i w_i [f_{i+}(1 - f_{i+})]$, while I_N is the total "inertia" corresponding to the ordination of the species by quadrats table (trace of the generalized variance-covariance matrix, G_T). See text for other denotation.

Re-scaling options	**Weighting options for quadrats, δ_a**	**Weighting options for species, w_i**	**Corresponding ordination method**	**Link with diversity indices**
(I) $x_{ai} = p_{ai}$	$\delta_a = 1/Q$	$w_i = 1$	Species-centered PCA*	
(II) $x_{ai} = \frac{p_{ai}}{\sqrt{V_i}}$ **	$\delta_a = 1/Q$	$w_i = 1$	PCA on the species correlation matrix*	$I_N=S$ (richness)
(III) $x_{ai} = p_{ai}/p_{a+}$	$\delta_a = 1/Q$	$w_i = 1$	Species-centered PCA on proportions (ter Braak 1983)	I_N = Simpson-Gini
(IV) $x_{ai} = p_{ai}/p_{a+}$	$\delta_a = p_{a+}$	$w_i = 1/p_{+i}$	Correspondence analysis (CA)*	$D = S$-1 (richness – 1)
		$w_i=log(1/p_{+i}) / (1-p_{+i})$	(Pélissier et al. 2003)	D = Shannon
		$w_i = 1$	Non-symmetric correspondence analysis (NSCA) (Gimaret-Carpentier et al. 1998)	D = SimpsonGini

* (Legendre and Legendre 1998); ** $V_i = \frac{1}{Q}\sum_a (p_{ai} - p_{+i})^2$

Figure 1

Spatial patterns shown by the main ordination axes provided by the application of correspondence analysis (CA) and non-symmetric correspondence analysis (NSCA) to vegetation data from the Counami Forest Reserve in French Guiana (7,189 trees of 59 species sampled in 411 quadrats). a) Generalized variogram for axis CA2 of the initial table (filled circles) and for the homologous axis from the residual table, after factoring out of 12 environmental categories (open circles). The dashed lines denote the 95% bilateral envelopes computed from 300 re-allocations of the specific composition to geographical locations (complete randomization for variograms from the initial data table and randomization within environmental categories for variograms from the residual table). The dotted line denotes the mean values for randomizations. b) Same as a) but for axis NSCA2. c) Same as a) but for CA3 (confidence envelopes are omitted for legibility, values within envelopes are marked by a square). d) Same as a) but for NSCA1. e) Generalized cross-variograms between the two main CA axes of the residual table. f) same as e) for NSCA.


CA
NSCA
a)
b)
c)
d)
e)
f)
$\gamma_{Gii}(h)$
$\gamma_{Gij}(h)$
Distance *h* (km)
Distance *h* (km)

Appendix for submission to the Electronic Ecological Archives :

matrix-algebraic presentation of the concepts and computations underlying the generalized, variogram-based multi-scale ordination

General denotation

Let **X** be a table expressing a measure of the abundance x_{ai} of S species (columns) within Q quadrats (rows). $\mathbf{x}_i$ and $\mathbf{x}_j$ are two columns of table **X**, relating to species i and j, respectively. Let **D** be a matrix containing quadrat weights (δ_a , $\sum_a \delta_a = 1$) on its main diagonal and zeros for all off-diagonal values, and let **W** be a S by S matrix containing the square root of species weights ($\sqrt{w_i}$) on its main diagonal and zeros outside. (In the main paper, Table 1 gives some options for abundance re-scaling and for quadrat and species weighting.)

Contiguity relationships

Let $\mathbf{L}_h$ be a Q by Q matrix expressing a contiguity relationship (*sensu* Lebart 1969) between the quadrats. For our variogram-based approach, we consider quadrats a and b as neighbors if the distance between the two is within the bounds of the distance class centered around h:

$\mathbf{L}_h(a,b)=1$ if $h_{a,b} \approx h$ and $\mathbf{L}_h(a,b)=0$ otherwise.

(A1)

To introduce quadrat weights into the analysis, we define the matrix $\mathbf{M}_h$ and the vector $\mathbf{E}_h$ such that:

$$\mathbf{M}_h = \mathbf{D}\mathbf{L}_h\mathbf{D} \quad \text{and} \quad \mathbf{E}_h = \mathbf{M}_h\mathbf{1}_Q$$

(A2)

where $\mathbf{1}$Q is the vector containing Q values equal to 1.

$\mathbf{M}_h$ contains, for each pair (a,b) of neighboring quadrats at "scale" h, the product $\delta_a\ \delta_b$ of their weights. $\mathbf{E}_h$ features, for each quadrat a, the sum of the weights of its neighbors multiplied by δ_a. Let $\mathbf{N}_h$ be the Q by Q matrix with $\mathbf{E}_h$ on its main diagonal and zeros elsewhere.

We shall assume that distance classes include all pairs of quadrats while being mutually exclusive. In such a case, the two following matrices:

$$\mathbf{M_T} = \sum_h \mathbf{M}_h \quad \text{and} \quad \mathbf{N_T} = \sum_h \mathbf{N}_h \tag{A2b}$$

are such that $\mathbf{M_T}$ is a Q by Q matrix that containing zeros on the diagonal while all values off the diagonal are equal to δ_a δ_b; $\mathbf{N_T}$ is a Q by Q matrix that containing $(1\text{-}\delta_a)\delta_a$ values on the diagonal and zeros elsewhere. With $\mathbf{M_T}$ and $\mathbf{N_T}$ it is as if each quadrat has all other quadrats as neighbors. Denoting $\mathbf{I}_Q$ the Q by Q diagonal identity matrix, we can also write:

$$\mathbf{N_T} = \mathbf{D(I}_Q - \mathbf{D)} \quad \text{and} \quad \mathbf{M_T} = \mathbf{D(1}_Q\mathbf{1}_Q{}^{\mathbf{t}} - \mathbf{I}_Q\mathbf{)D}$$

(A3)

(where the exponent ' t ' is the matrix transpose). Thus:

$$\mathbf{N_T} - \mathbf{M_T} = \mathbf{D} - \mathbf{D1}_Q\mathbf{1}_Q{}^{\mathbf{t}}\mathbf{D}$$

(A4)

Equivalent expressions of the generalized variance-covariance matrix

Let $\mathbf{G_T}$ be the generalized variance-covariance matrix, irrespective of distance classes, that can be directly computed from table **X** using weighting options for rows and columns defined by matrices **D** and **W**, respectively. $\mathbf{G_T}$ contains, for each species couple (i,j), the generalized covariances, g_{ij} as defined by Eq.1 and Eq.2 in the main paper:

$$g_{ij} = \sum_{a,b} g_{ij}(a,b)$$

(A5)

Usual algebraic manipulations allow us to re-write Eq.1 and Eq.A5 as:

$$g_{ij} = \sqrt{w_i w_j}\left(\sum_a^Q \delta_a x_{ai} x_{aj} - \bar{x}_i \bar{x}_j\right) \tag{A6}$$

where $\bar{x}_i$ and $\bar{x}_j$ are the **D**-weighted means of $\mathbf{x}_i$ and $\mathbf{x}_j$, respectively.

($\bar{x}_i = \sum_a^Q \delta_a x_{ia}$ or, equivalently, $\bar{x}_i = \mathbf{X}_i{}^t\mathbf{D1}_Q$)

The matrix expression of g_{ij} is thus:

$$g_{ij} = \mathbf{W(x}_i{}^t\mathbf{Dx}_j - \mathbf{x}_i{}^t\mathbf{D1}_Q\mathbf{x}_j{}^t\mathbf{D1}_Q\mathbf{)W}$$

(A7)

which generalizes into:

$$\mathbf{G_T} = \mathbf{W(X}^t\mathbf{DX} - \overline{\mathbf{X}}^t\mathbf{D}\overline{\mathbf{X}}\mathbf{)W}$$

(A8)

where $\overline{\mathbf{X}} = \mathbf{1}_Q\mathbf{1}_Q{}^t\mathbf{D}\mathbf{X}$ and where $\overline{\mathbf{X}}{}^t\mathbf{D}\overline{\mathbf{X}} = [\bar{x}_i\bar{x}_j]$

(A9)

Note that we may also write:

$$\mathbf{G}_T = \mathbf{W}(\mathbf{X} - \overline{\mathbf{X}})^t\,\mathbf{D}(\mathbf{X} - \overline{\mathbf{X}})\mathbf{W} \qquad \text{(A10)}$$

On the other hand**,** it is important to note that $\mathbf{G}_T$ can be directly computed as:

$$\mathbf{G}_T = \mathbf{W}\mathbf{X}^t(\mathbf{N}_T - \mathbf{M}_T)\mathbf{X}\mathbf{W}$$

(A11)

Proof of Eq.A11:

$\mathbf{X}^t(\mathbf{N}_T - \mathbf{M}_T)\mathbf{X} = \mathbf{X}^t(\mathbf{D} - \mathbf{D}\mathbf{1}_Q\mathbf{1}_Q{}^t\mathbf{D})\mathbf{X}$ (using Eq.A4)

$\mathbf{X}^t(\mathbf{D} - \mathbf{D}\mathbf{1}_Q\mathbf{1}_Q{}^t\mathbf{D})\mathbf{X} = \mathbf{X}^t\mathbf{D}\mathbf{X} - \mathbf{X}^t\mathbf{D}\overline{\mathbf{X}}$ (using Eq.A9)

Noting that $\mathbf{X}^t\mathbf{D}\overline{\mathbf{X}} = \overline{\mathbf{X}}{}^t\mathbf{D}\overline{\mathbf{X}}$, allows us to write:

$$\mathbf{X}^t(\mathbf{N}_T - \mathbf{M}_T)\mathbf{X} = \mathbf{X}^t\mathbf{D}\mathbf{X} - \overline{\mathbf{X}}{}^t\mathbf{D}\overline{\mathbf{X}}$$

(A12)

Partition of the generalized variance-covariance matrix among distance classes

The very definition of matrices $\mathbf{N}_T$ and $\mathbf{M}_T$ (Eq.A2b), along with Eq.A11, enables partition of $\mathbf{G}_T$ into strictly additive components, $\mathbf{G}_h$, that relate each to a distance class:

$$\mathbf{G}_T = \sum_h \mathbf{G}_h = \sum_h \mathbf{W}\mathbf{X}^t(\mathbf{N}_h - \mathbf{M}_h)\mathbf{X}\mathbf{W} \qquad \text{(A13)}$$

$\mathbf{G}_h$ is the generalized variance-covariance matrix defined for the distance class h by the neighboring relationship expressed by the matrices $\mathbf{N}_h$ and $\mathbf{M}_h$. $\mathbf{G}_h$ translates easily into generalization of Wagner's variogram matrix (2003) by a division of all its values by

$$K(h) = \sum_{a,b\,|\,h_{ab}\approx h} \delta_a\delta_b \qquad \text{or} \quad K(h) = \mathbf{1}_Q{}^t\mathbf{M}_h\mathbf{1}_Q$$

(A14)

Equations A2, A13 and A14 are used for easy programming of the method as well as efficient computations via any matrix-oriented programming environment, as we did with Matlab® and R (Ihaka and Gentleman 1996): see the freely available library "msov" on http://pbil.univ-lyon1.fr/CRAN/.)

For a particular species couple i and j we obtain:

$$g_{ij}(h) = \mathbf{W}\mathbf{x}_i{}^t(\mathbf{N}_h - \mathbf{M}_h)\mathbf{x}_j\mathbf{W} \qquad \text{(A15)}$$

Dividing by the scaling factor *K(h)* gives the value at "scale" *h* of the generalized version of either cross-variogram (*i≠j*) or variogram (*i=j*) :

$$\gamma g_{ij}(h) = \frac{1}{K(h)} g_{ij}(h) \qquad \text{(A16)}$$

Multi-scale ordination

All the ordination methods mentioned in Table 1 of the main paper are based on the singular values decomposition (svd) of the appropriate version of $\mathbf{G_T}$ to compute eigenvectors, $\mathbf{u}_f$, and associated eigenvalues, λ_f. Let $\mathbf{U}_f$ be the matrix having all the eigenvectors $\mathbf{u}_f$ as columns and let $\mathbf{\Lambda}$ be the diagonal matrix having the eigenvalues λ_f on its diagonal. Both eigenvectors and eigenvalues of $\mathbf{G_T}$ can be partitioned by distance classes:

$$\mathbf{F}_h = \mathbf{U}_f{}^t \mathbf{G}_h \mathbf{U}_f \text{ and } \lambda_f(h) = \mathbf{u}_f{}^t \mathbf{G}_h \mathbf{u}_f \qquad \text{(A17)}$$

$\mathbf{F}_h$ is the variance-covariance matrix of the eigenvectors at scale *h*. Scale-dependent variogram/cross-variogram matrices of the eigenvectors are deduced by the appropriate scaling (Eq. A16). Note also that:

$$\sum_h \mathbf{F}_h = \mathbf{U}_f{}^t \left(\sum_h \mathbf{G}_h \right) \mathbf{U}_f = \mathbf{U}_f{}^t \mathbf{G_T} \mathbf{U}_f = \mathbf{\Lambda} \qquad \text{(A18)}$$

Taking environmental heterogeneity into account

Let us now suppose that a table, **Z**, containing assessments of *P* environmental variables for the *Q* quadrats, is available in addition to table of species composition. It is well established that the centered by columns table, $\mathbf{X_C}$, may be partitioned into an approximated table,

$$\mathbf{A_C} = \mathbf{Z}(\mathbf{Z}^t\mathbf{D}\mathbf{Z})^{-1}\mathbf{Z}^t\mathbf{D}\mathbf{X_C}$$

(A19)

and a residual table, $\mathbf{R_C} = \mathbf{X_C} - \mathbf{Ac}$ (Sabatier et al. 1989).

In the same manner, it may also have a direct decomposition of the initial table **X**:

$$\mathbf{A} = \mathbf{Z}(\mathbf{Z}^t(\mathbf{N_T} - \mathbf{M_T})\mathbf{Z})^{-1}\mathbf{Z}^t(\mathbf{N_T} - \mathbf{M_T})\mathbf{X} \text{ and } \mathbf{R} = \mathbf{X} - \mathbf{A}$$

(A20)

After factoring out the environmental variables, residual spatial patterns may be studied by the multi-scale analysis of spatial covariances derived from table **R** or $\mathbf{R_C}$. The total residual variance-covariance matrix, $\mathbf{G_{RT}}$, is computed as:

$$\mathbf{G_{RT}} = \mathbf{W}\mathbf{R}^t(\mathbf{N_T} - \mathbf{M_T})\mathbf{R}\mathbf{W} = \mathbf{W}\mathbf{R_C}^t(\mathbf{N_T} - \mathbf{M_T})\mathbf{R_C}\mathbf{W} \qquad \text{(A21)}$$

and is broken down with respect to distance classes:

$$\mathbf{G}_{\mathbf{R}h} = \mathbf{W}\mathbf{R}^t(\mathbf{N}_h - \mathbf{M}_h)\mathbf{R}\mathbf{W} = \mathbf{W}\mathbf{R}_\mathbf{C}{}^t(\mathbf{N}_h - \mathbf{M}_h)\mathbf{R}_\mathbf{C}\mathbf{W}$$

(A22)

The additive partitioning of $\mathbf{G}_{\mathbf{RT}}$ with respect to distance classes thus enables an investigation of the residual spatial patterns by a multi-scale ordination scheme analogous to that defined by Eq.A17 and A18.

Literature cited in this appendix

Ihaka, R., and R. Gentleman. 1996. R: a language for data analysis and graphics. Journal of Computational and Graphical Statistics **5**:299-314.

Lebart, L. 1969. Analyse statistique de la contiguïté. Publications de l'Institut de Statistiques de l'Université de Paris **28**: 81-112.

Sabatier, R., J.-D. Lebreton, and D. Chessel. 1989. Principal component analysis with instrumental variables as a tool for modelling composition data. Pages 341–352 *in* R. Coppi and S. Bolasco, Editors. Multiway data analysis, Elsevier Science Publishers, Amsterdam, The Netherlands.

Wagner, H.H. 2003. Spatial covariance in plant communities: integrating ordination, geostatistics, and variance testing. Ecology **84**: 1045-1057.

3. Comparing and classifying one-dimensional spatial patterns: an application to laser altimeter profiles

auteurs

Sébastien Ollier
Daniel Chessel
Raphaël Pélissier
Pierre Couteron
Jean Thioulouse

résumé

Numerical analyses of remotely sensed data may valuably contribute to an understanding of the vegetation/land surface interface by pointing out at which scales a given variable displays a high level of spatial variability. Thus, there is a need of methods aimed at classifying many one-dimensional signals, such as airborne laser profiles, on the basis of their spatial structure. The present paper proposes a theoretical framework ensuring a consistent combination of a multi-scale pattern characterization, based on the Haar wavelet variance (also called in ecology Two Terms Local Variance, TTLV), with two multivariate techniques such as principal components analysis (PCA) and hierarchical cluster analysis. We illustrate our approach by comparing and classifying 257 laser profiles, with a length of 64 measurements (448 m), that were collected by the BRGM in French Guiana over three main landscape units with distinct geomorphological and ecological characteristics. We calculate for each profile a scalogram that summarized the multi-scale pattern and analyze the structural variability of profiles via a typology and a classification of one-dimensional patterns. More than 80% of the variability between spatial patterns of laser profiles has been summarized by two PCA axes, while four classes of spatial patterns were identified by cluster analysis. Each landscape unit was associated with one or two dominant classes of spatial patterns. These results confirmed the ability of the method to extract landscape scaling properties from complex and large sets of remotely sensed data.

mots-clés

Multi-scale pattern analysis, Haar wavelet variance, Two Terms Local Variance, Classification of spatial patterns, Laser altimeter profile, Landforms, Forest canopy, Tropical rain forest

journal

Remote Sensing of Environment **85** (2003) 453-462

Introduction

Concepts of spatial pattern and scale play a central role in the study of both ecological and land surface processes (Scheidegger, 1991; Levin, 1992). The uneven distribution through space of resource, material and biomass is, indeed, both a determinant and a result of dynamic processes taking place at the interface between vegetation and landforms. Biotic as well as abiotic processes generally exert an influence on a broad range of scales, which does not mean that all scales are equally relevant to the study of a given phenomenon or land system. Hence, pattern characterization via numerical analyzes of remotely-sensed data may valuably contribute to an understanding of the vegetation/land surface interface by pointing out at which scales a given variable (*e.g.* total biomass) displays a high level of spatial variability.

Several methods, using autocorrelation, fractals, variograms, or wavelet variance have been proposed to investigate scales of spatial heterogeneity in remotely-sensed data (Pachepsky et al., 1997; Rango et al., 2000; Parker et al., 2001). All these methods characterize a pattern observed in a particular sampling unit, for instance along a given laser transect or within a portion of a satellite scene, by studying the relationship between variance and scale (Ver_Hoef et al., 1993; Dale, 1999; Dale et al., submitted). What has been largely missing, until now, is a general approach enabling multi-scale comparisons between a large number of spatial patterns, *i.e.*, between many sampling units in which patterns are observed and quantified via one of the above methods. A reciprocal question would be how to assess the relative importance of scales on the basis of a large set of sampling units.

This is the very scope of the present paper which proposes a theoretical framework for a systematic comparison of spatial patterns observed in one-dimensional sampling units (*i.e.* transects). By revisiting an established method for pattern quantification, namely the Haar wavelet variance (Bradshaw & Spies, 1992; Burrus et al., 1998; Dale & Mah, 1998)- also known in ecological sciences as Two-Terms-Local-Variance (Hill, 1973)-, we propose a multivariate approach allowing to consistently compare and classify one-dimensional spatial patterns on a multi-scale basis. We shall also consider an application of this approach to airborne laser profiles obtained for a tropical forested landscape in French Guiana. The objective of that application is to compare patterns of laser altimeter profiles with the geomorphological and ecological context to determine whether pattern analysis of laser data could be a useful means for quantifying scaling properties of tropical forested landscapes.

Methods

Quantifying patterns via the Haar wavelet variance

Wavelet transforms are becoming increasingly popular for signal analysis, denoising and compressing (Burrus et al., 1998). One way to produce a wavelet transform, $W(b,x_i)$, from one-dimensional $y=[y_i]_{1\le i\le n}$, is to compare the data sequence with a particular template or wavelet function, $w(t)$, centred at successive locations x_i, and expressing the scale, *b*, of the analysis (Bradshaw & Spies, 1992; Dale & Mah, 1998):

$$W(b,x_i)=\frac{1}{\sqrt{b}}\sum_{j=1}^{n}y_j w\left(\frac{x_j-x_i}{b}\right).$$

This formulation corresponds to a continuous wavelet transform (Misiti et al., 1993), for which the analyzing template is shifted smoothly over the full signal. Wavelet transforms generate results in terms of scale and position, *i.e.*, a two-dimensional array from a one-dimensional transect. To facilitate comparisons between transects, the more synthetic wavelet variance function, $V(b)$, can be computed by averaging all squared values $W(b,x_i)^2$ obtained at a particular scale *b*. In so doing, emphasis is put on scales while analytical information related to position is dropped to enable a systematic comparison of spatial patterns observed in many sampling units

In ecological sciences, wavelet variances have long been used to analyse single transects under the name of Two Terms Local Variance or TTLV (Hill 1973). However, Dale & Mah (1998) recognized that the TTLV can be seen as a variance function of the oldest and simplest wavelet template, which was due to Haar (1910). With this template we have:

$$\begin{cases} w(t)=1 \text{ if } \left(0\le t<\frac{1}{2}\right) \\ w(t)=\text{-}1 \text{ if } \left(\frac{1}{2}\le t<1\right) \\ w(t)=0 \text{ otherwise} \end{cases}$$

Thus, using the Haar wavelet at a particular scale *b*, is equivalent to computing the difference between successive blocks of *b* values observed along the transect. The wavelet variance is then computed from the average of the squared difference between blocks, which can be written for a finite data sequence, $y=[y_i]_{1\le i\le n}$ of *n* values, as:

$$V(b)=average\left(W(b,x_i)^2\right)=\frac{1}{K(b)}\sum_{i=1}^{n+1-2b}\left[\sum_{j=i}^{i+b-1}(y_j-y_{j+b})\right]^2$$

where *K(b)* is a scaling coefficient. Its choice may depend on the objective of the analysis. Our aim being to achieve a multi-scale comparison of spatial patterns, there is a need of a standardization of *V(b)* values, which would allow consistent comparisons of pattern intensity for different values of *b*. Indeed, large scales features (*e.g.*, landforms) are likely to have a higher variance than the features at smaller scales (*e.g.*, emanating from forest canopy), thereby determining most of the results of a multi-scale pattern comparison if *a priori* standardization is omitted.

Using matrix formulation to standardize *V(b)* values

V(b) is a quadratic form and, as such, can be written as: $V(b) = \frac{\mathbf{y}^t \mathbf{A_b} \mathbf{y}}{K(b)}$ where $\mathbf{y}$ stands for the vector of observations and where $\mathbf{A_b}$ denotes the square matrix (or metric) defining the quadratic form at scale *b*. This matrix is easily computable on the basis of the relationships linking observational units and blocks. Let $\mathbf{P}$ be the matrix expressing the affectation of the *n* observational units to the $N_b = n+1-b$ blocks defined at scale *b*. $\mathbf{P}(l,m)$ is one if unit *l* belongs to block *m* and zero otherwise. Let $\mathbf{M}$ stand for the $N_b \times N_b$ matrix expressing the neighbourhood relationship between blocks. Only adjacent blocks that do not overlap are considered as neighbours, for instance blocks $[x_i,\ x_i + b\text{-}1]$ and $[x_i + b, x_i + 2b\text{-}1]$. $\mathbf{M}(l,m)$ is one if blocks *l* and *m* are neighbours and zero otherwise. Finally, let $\mathbf{N}$ be the diagonal matrix, with $\mathbf{N}(m,m)$ being the number of neighbours of block *m* and with $\mathbf{N}(l,m) = 0$ for $l \neq m$. Matrix $\mathbf{A_b}$ can be rewritten as $\mathbf{A_b} = \mathbf{P^t(N - M)P}$ (Lebart, 1969; see appendix for an example).

Note that the sum of all diagonal terms of $\mathbf{A_b}$, i.e., trace($\mathbf{A_b}$) is $2b(n+1-2b)$, a value which has been proposed by Hill (1973) for standardizing the TTLV. Indeed, taking *K(b)*=trace($\mathbf{A_b}$) (Euclidean standardization) allows to consider *V(b)* as an estimate of variance at scale *b*. However, dividing by trace($\mathbf{A_b}$) does not ensure that *V(b)* values are comparable at different scales. Furthermore, matrices $\mathbf{A_b}$ are not diagonal matrices due to the use of overlapping blocks.

This problem can be solved using what has been defined by Chatelin (1988) as a "spectral standardization", which relies on a singular value decomposition (SVD) of $\mathbf{A_b}$. (All matrices $\mathbf{A_b}$ are semi-definite positive matrices and can be decomposed by SVD.) With this approach, the spatial pattern defined by the first eigenvector of $\mathbf{A_b}$ is taken as reference for

pattern analysis at scale *b*, and the associated eigenvalue, $\lambda_1(b)$, is used as reference variance for standardization, *i.e.*, $K(b)=\lambda_1(b)$. For instance, considering a transect length of n=64 and scales of $b=\{1,2,4,8,16,24\}$ yields the reference spatial patterns displayed in Fig 1. Using these eigenvectors as references means that, for a given transect, pattern intensity at a particular scale *b*, will be equal to 1 if and only if the transect values, $y_{1\le i\le n}$, are proportional to the reference pattern. Thus, comparing transects on the basis of their underlying multi-scale structure, as well as assessing the relative importance of scales on the basis of a large set of transects can be both conceived in a consistent way.

Pattern ordination and classification

Let us consider the $p\times q$ table $\mathbf{Z}$, for which each of the p rows represent the *V(b)* values of a given transect (the latter are called "scalogram" *sensu* Strang & Nguyen, 1996), and each of the q columns contain the scalogram values related to a given scale $b\in\left\{1,2,3,...,\frac{n}{2}\right\}$. Thus, in this table, the transects correspond to statistical units, while scales (*i.e.*, sizes of blocks) are quantitative variables characterizing the transects in terms of spatial patterns. Note that Couteron (2002) used an analogous table for texture-based comparisons of air photographs, with the difference that Fourier spectra were used instead of Haar scalograms. Fourier spectra do not require any standardization prior to comparison, since reference functions (sine and cosine) are orthogonal, but Fourier decomposition is reputed to be less robust to non-stationarity than its wavelet analogues (Burrus et al., 1998).

We submit the $(p\times q)$ table $\mathbf{Z}$ to a Principal Components Analysis (PCA) on correlation matrix (Manly, 1994) that looks for a limited number of synthetic new variables accounting for a substantial share of the variability between scalograms. These new variables are linear combinations of initial variables and are called principal components. As scalograms have been preliminarily standardized by the "spectral" norm, each of the scale variable has the same weight in the analysis. PCA results have been displayed using a particular graphical method called biplot (Gabriel, 1981).

As a complement to PCA ordination, a hierarchical cluster analysis has been carried out to classify scalograms. The analysis used the Euclidean distance between two scalograms *i* and *j*, $d_{ij}^{\,2}=\sum_{k=1}^{q}\left(z_{ik}-z_{jk}\right)^2$, which is consistent with PCA, and the Ward's minimum variance criterion that yields compact spherical clusters (Gordon, 1981).

All computations involved in the preparation of this paper were carried out using R (Ihaka & Gentleman, 1996), ADE-4 (Thioulouse et al., 1997) and ArcView® as softwares, with both pre-programmed and personal routines.

Application to laser data

To illustrate the method, we used airborne laser altimetry data extracted from a very large data set of ca. 10^5 km of cumulated laser profiles, that were recorded in 1996 throughout French Guiana by the BRGM (Delor et al., 1998).

Study site

We have restricted ourselves to a study site of 15,000 ha located at about 143 km towards the northwest of the main-town Cayenne, between 5°28' and 5°38' North latitude and -53°17 and -53°28' West longitude (Fig. 2). It is situated in an unlogged rain forest called Counami forest. Climate is wet tropical with annual rainfall ranging between 2,750 and 3,000 mm and scattered over 9 months (Blancaneaux, 2001).

Within the Counami site, experienced geomorphologists have identified three main landscape units (Fig. 2) that differ in altitude and morphological complexity (Hutter, 2001). The first unit, *i.e.*, "alluvial plains" (A), is characterized by a very simple and flat relief, and encompasses the valleys of the three main rivers (Counamama, Counami and Iracoubo) flowing through the study site. The second unit (B) stands for complex relief forms, having maximal altitudes below 60 m above sea level, and corresponding to tabular landforms that gently slopes toward talwegs. The last unit (C) is characterized by an altitude generally exceeding 60 m above see level, associated with complex landforms that present steep slopes with small round-off summits.

Laser data

An airborne laser altimeter was used to measure the distance from airplane to landscape surface. The profiling laser altimeter was a pulsed gallium-arsenide diode laser operating at a frequency of 195 Hz and a wavelength of 905 nm. The field of view of the laser was 5 radians giving a "footprint" on the ground equal to 12 cm for the nominal altitude used during these flights. The timing mechanism in the laser receiver enabled a vertical resolution of 10 cm for a single measurement. The altitude of the airplane was approximately 120 m above the forest canopy with a nominal ground speed of 260 km/h. Data were pre-processed during recording by combining the number of shots so that a laser measurement occurs at intervals of 7 m along the flight line (Fig. 2). The geo-location of laser footprint was established with a spatial

resolution of 2-3 m using a combination of ancillary data recorded simultaneously to laser measurements (video frames and GPS data). Landscape surface elevation was finally calculated for each measurement by using ground elevations along a flight line (GPS data) to convert the relative data into absolute elevations. Each measurement relates to the first obstacle encountered by the laser, and may correspond to vegetation items (leaf, branch) or to the forest floor itself.

Flight lines were oriented 30° N and separated by 500m. We extracted a set of $p = 556$ transects with a length of $n = 64$ points of measurements (448 m) from the 30 flight lines that intersected the study site. Each transect was represented in the geographical space by a point corresponding to its middle (Fig. 2). We eliminated transects that intersected several geomorphological types and kept only $p = 257$ homogeneous transects. We computed the scalogram for each of these 257 transects, using the following set of block sizes $b \in \{1, 2, 4, 6, 8, 12, 16, 24\}$. Then the scalograms were compared and classified to provide a typology of transects based on their spatial pattern.

Results

The two main PCA axes accounted for 81% of the variability between scalograms, with 48% and 33% for the first and the second axes, respectively (Fig. 3a). The first axis displayed a high negative correlation with small block sizes and a high positive correlation with large block sizes (Fig. 3b). The second axis was negatively correlated with intermediate block sizes. In terms of transect scores (Fig. 3b), it was particularly difficult to define clusters of transects by visual delineation. However, the hierarchical cluster analysis yielded a dendrogram containing four well-defined clusters (Fig. 4).

Centers of gravity of the clusters identified through the previous dendrogram are represented on the biplot in Figure 3b. For each cluster, the average scalogram is displayed in Figure 5a along with the laser profile having the smallest Euclidian distance to the average scalogram of the cluster. Cluster 1 was typical of fine-grained patterns, with scalograms that were exclusively skewed towards small scales. Laser profiles belonging to that cluster displayed a substantial variability, that was mainly observable along axis 2 in Figure 3b, with a gradation of small scales from b=1,2 (transect 551, 379) to b=4 (transect 360) and 6 (transect 189), i.e. from structures of size 7 to 42 m. As opposed to that cluster, the second one (cluster 2) was characterized by patterns which mainly presented large-scale variation between 100 m (b=16; transects 152, 346) and 200 m (b=24; transects 181, 297). Scale variability within this cluster was mainly apparent along axis 2. Cluster 3 gathered patterns

characterized by the importance of both small and large scales (i.e. <40 m and >100 m, respectively). It was typically a class of multi-scale patterns. Intensity attached to small scales was quite lower for this cluster than for cluster 1. Cluster 4 featured all the patterns characterized by the dominance of intermediate scales, ranging from about 40 to 100 m.

Laser transects belonging to the same cluster were not evenly distributed throughout the study area (Fig.6). In particular, transects belonging to clusters 1 and 2 appeared clearly aggregated. This means that neighbouring transects displayed similar spatial patterns of laser response. The local heterogeneity of landscape surface roughness was thus quite limited. Moreover, laser patterns were related to landscape units, since the contingency table based on the cross-classification of the three landscape units and the four clusters of patterns (Table 1) departed significantly from the null hypothesis of independence (chi2 test, P=2e-12). Indeed, each landscape unit had most of its transects belonging to a particular cluster. Patterns defining cluster 1 were generally found in the alluvial plain (landscape unit A), while patterns making clusters 3 and 4 characterized the landscape unit B, and patterns of cluster 2 mostly belonged to landscape unit C. The differences between landscape units were thus explained simultaneously by large and small scales patterns. Large scale differences were not surprising since they reflected large scale topographical fluctuations that characterized the landscape units B and C . The scale gradient observed from the laser profiles, that stretched from the alluvial plain to the complex landforms was also consistent with the visual aspect of the radar scenes (see Fig. 2c). Results at smaller scales (large intensity for the alluvial plain, lower intensity for the complex forms) were less expected, but may be interpreted as differences in forest cover: small-scaled structures may correspond to convex units of canopy surface topography formed by the crown of emergent trees, isolated or juxtaposed with a minimal inter-crown space (Pitman et al., 1999; Birnbaum, 2001). Other forms of forest organization can result in large-scale variations due to the presence of canopy gaps (Bradshaw & Spies, 1992; Birnbaum, 2001). These forms of canopy organization are liable to vary in relation to topographical and geological factors. Moreover, distinct floristic compositions were found for parts of the Counami forest corresponding roughly to the three landscape units (Couteron et al., 2002). Here, the correspondence between landscape units and clusters remained fuzzier, because the variability of laser profiles within landscape units was large (Table 1). For example, in the alluvial plain, in spite of a clear dominance of fine-grained patterns (cluster 1), large gaps in the forest cover may occasionally determine the presence of laser profiles belonging to the other clusters.

Discussion

Using laser data, we have illustrated a new method for classifying one-dimensional patterns. In spite of a coarse resolution of the laser signal, results regarding geomorphology and to some extent forest canopy structure are interesting and non trivial since they were related to an independent mapping of landforms. This is in agreement with results of several studies that have demonstrated the usefulness of various kinds of laser data to characterize vegetation cover and landforms (Pachepsky et al., 1997; Pachepsky & Ritchie, 1998; Drake & Weishampel, 2000; Weishampel et al., 2000), and to investigate scaling properties of landscapes. Indeed as for other studies, we have found spatial patterns expressing themselves at several distinct scales. Of course, the characteristic of the laser information that was used did not allow a clear distinction of landform *vs.* canopy cover structure at intermediate scales. But other kinds of laser techniques can allow such a distinction thanks to a higher spatial resolution, or to new sensors like the LVIS generation (Blair et al., 1999), that yield separate signals for canopy cover and forest floor. Corresponding data sets will, nevertheless, represent a large amount of information that will increase the need of methods allowing consistent comparison and classification of spatial patterns. Thus, whatever the limitation of the data set we used for illustration, the present study demonstrated the efficiency of our new method to summarize and classify a large number of laser profiles.

Indeed, the method was able to summarize a rich set of data into a limited number of ordination axes that are linear combinations of spatial scales. More than 80% of the variability between scalograms, and thus between spatial patterns, was accounted for by the two main PCA axes. As laser data contain pertinent information on landscape scaling properties, both the main axes of the PCA and the cluster dendrogram constitute valuable tools to explore, summarize and monitor landforms and/or vegetation covers. This kind of analysis is actually multi-scale at least for the range of scales that can be considered via a sampling window having a particular length (here 448m).

Our approach is not restricted to a particular method of spatial pattern quantification, since pattern classification may stem from any method that produce an expression of variance with respect to scale. Indeed, the variance at a particular scale, *b*, can always be represented as a quadratic form ($\mathbf{x}^t\mathbf{A}_b\mathbf{x}$) of a data vector $\mathbf{x}$. Consequently, the analysis of matrices $\mathbf{A}_b$, via singular value decomposition, provides explicit representations of the reference spatial patterns that a given method consider at a given scale (Fig. 1). This opens the way to a standardization of variance *vs.* scale functions (scalograms).

In spite of this unifying principle, the extent to which the choice of a particular method of pattern quantification (via a family of matrices Ab), may have some influence on the final result of the pattern classification is still an open question. For the present paper we used the Haar wavelet variance (also known as Two Terms Local Variance) which is closely related to the variogram (Ver_Hoef et al., 1993). The use of other simple wavelet templates (e.g. the 'French Top Hat'; Dale & Mah 1998) is also thinkable. Another alternative is Fourier decomposition, which has been already used for classifying spatial patterns that display a strong periodic component (Couteron 2002). Since our analysis only deals with scales and not with positions, the advantage of wavelets over Fourier decomposition may, in fact, be questioned. From a theoretical standpoint, it is well-known that Fourier analysis is very good at detecting genuine periodicity, though non-stationarity may have some blurring effects. Taking short portions of the signal, as our 'transects' of 64 observations, may limit non-stationarity, but is also detrimental to the resolution of the Fourier spectrum (Kumaresan, 1993). Hence, even if only scales are dealt with, using wavelet alternatives to Fourier decomposition retain a substantial appeal if one has *no a priori* reason to suspect the presence of periodic features in the data. From a practical standpoint, however, the sensibility of a pattern classification to the Fourier vs. Haar wavelet choice is a question which is still largely to be explored from both real-world and computer-generated data. This is, anyway, beyond the scope of the present paper, which intend mainly to underline that, thanks to spectral standardization, consistent multi-scale comparisons of one-dimensional patterns are possible from various families of scale vs. variance functions.

Acknowledgements

This work was undertaken thanks to a research agreement between the "Bureau de Recherches Géologiques et Minières" (BRGM) and the “Institut de Recherche pour le Développement” (IRD) that allowed Sébastien Ollier to use the laser data from the 1996 airborne geophysical project of BRGM in French Guiana. We are particularly grateful to Catherine Truffert from BRGM-Orléans, who actively supported this collaboration.

Appendix

The goal of this appendix is to illustrate on an example how the wavelet variance function at scale *b*, $V(b)=\frac{1}{K(b)}\sum_{i=1}^{n+1-2b}\left[\sum_{j=i}^{i+b-1}(y_j-y_{j+b})\right]^2$, can be rewritten under a quadratic form $V(b)=\frac{\mathbf{y^t A_b y}}{K(b)}$ where $\mathbf{A_b = P^t(N-M)P}$.

Let us consider a transect $\mathbf{y^t}=(y_1,y_2,y_3,y_4,y_5,y_6)$ of length *n=6* , and a block size *b=2*.

There are $N_2=n+1-b=5$ blocks of length 2 in the transect: $[(y_1,y_2);(y_2,y_3);(y_3,y_4);(y_4,y_5);(y_5,y_6)]$. This information is summarized in the matrix $P_{(N_2,n)}=\begin{pmatrix}1&1&0&0&0&0\\0&1&1&0&0&0\\0&0&1&1&0&0\\0&0&0&1&1&0\\0&0&0&0&1&1\end{pmatrix}$. The neighbourhood relationships between blocks is expressed by the matrix $M_{(N_b,N_b)}=\begin{pmatrix}0&0&1&0&0\\0&0&0&1&0\\1&0&0&0&1\\0&1&0&0&0\\0&0&1&0&0\end{pmatrix}$ and the number of neighbours of each blocks is represented on the diagonal of the matrix $N_{(N_b,N_b)}=\begin{pmatrix}1&0&0&0&0\\0&1&0&0&0\\0&0&2&0&0\\0&0&0&1&0\\0&0&0&0&1\end{pmatrix}$. The Haar wavelet variance at scale , *b=2*, is

$$V(2)=\frac{\left[(y_1+y_2-y_3-y_4)^2+(y_2+y_3-y_4-y_5)^2+(y_3+y_4-y_5-y_6)^2\right]}{K(b)}=\frac{\begin{pmatrix}y_1+y_2\\y_2+y_3\\y_3+y_4\\y_4+y_5\\y_5+y_6\end{pmatrix}^t\begin{pmatrix}1&0&-1&0&0\\0&1&0&-1&0\\-1&0&2&0&-1\\0&-1&0&1&0\\0&0&-1&0&1\end{pmatrix}\begin{pmatrix}y_1+y_2\\y_2+y_3\\y_3+y_4\\y_4+y_5\\y_5+y_6\end{pmatrix}}{K(b)},\ \textit{i.e.}$$

$$V(2)=\frac{(\mathbf{Py})^t(\mathbf{N-M})(\mathbf{Py})}{K(b)}=\frac{\mathbf{y^t(P^t(N-M)P)y}}{K(b)}=\frac{\mathbf{y^tA_by}}{K(b)}\qquad\text{with}\qquad\mathbf{A_b=P^t(N-M)P}.$$

References

Birnbaum, P., 2001. Canopy surface topography in a French Guiana forest and the folded forest theory. Plant Ecology, 123: 293-300.

Blair, J.B., Rabine, D.L., & Hofton, M.A., 1999. The Laser Vegetation Imaging Sensor: a medium-altitude, digitisation-only, airborne laser altimeter for mapping vegetation and topography. ISPRS Photogrammetry and Remote Sensing, 54: 115-122.

Blancaneaux, P., 2001. Le climat guyanais. In: J. Barret (Editor), Atlas illustré de la Guyane. CNES, IESG, IRD, Région Guyane, pp. 46-49.

Bradshaw, G.A. & Spies, T.A., 1992. Characterizing canopy gap structure in forests using wavelet analysis. Journal of Ecology, 80: 205-215.

Burrus, C.S., Gopinath, R.A., & Guo, H., 1998. Introduction to Wavelets and Wavelet Transforms. Prentice-Hall, Upper Saddle River, NJ.

Chatelin, F., 1988. Valeurs propres de matrices. Masson, Paris.

Couteron, P., 2002. Quantifying change in patterned semi-arid vegetation by Fourier analysis of digitised aerial photographs. International Journal of Remote Sensing.

Couteron, P., Pélissier, R., Mapaga, D., Molino, J.F., & Teillier, L., 2002. Ecological valorisation of a management-oriented forest inventory in French Guiana. Forest Ecology and Management.

Dale, M.R.T. & Mah, M., 1998. The use of wavelets for spatial pattern analysis in ecology. Journal of Vegetation Science, 9: 805-814.

Dale, M.R.T., 1999. Spatial pattern analysis in plant ecology. Cambridge University Press, Cambridge.

Dale, M.R.T., Dixon, P., Fortin, M.J., Legendre, P., Myers, D., & Rosenberg, M.S., submitted. The conceptual and mathematical relationships among methods for spatial analysis. Ecography.

Delor, C., Perrin, J., Truffert, C., Asfirane, F., & Rossi, P., 1998. Images géophysiques dans le socle guyanais. Géochronique, 67: 7-12.

Drake, J.B. & Weishampel, J.F., 2000. Multifractal analysis of canopy height measures in a longleaf pine savanna. Forest Ecology and Management, 128: 121-127.

Gabriel, K.R., 1981. Biplot display of multivariate matrices for inspection of data and diagnosis. In: V. Barnett (Editor), Interpreting multivariate data. John Wiley and Sons, New York, pp. 147-174.

Gordon, A.D., 1981. Classification. Chapman and Hall, London.

Hill, M.O., 1973. The intensity of spatial pattern in plant communities. Journal of Ecology, 61: 225-235.

Hutter, S., 2001. Etude geomorphologique du massif forestier de Counami. Technical report, CIRAD.

Ihaka, R. & Gentleman, R., 1996. R: a language for data analysis and graphics. Journal of Computational and Graphical Statistics, 5: 299-314.

Kumaresan, R. 1993. Spectral analysis. In: S.K. Mitra & J.F. Kaiser (Editors), Handbook for digital signal processing . John Wiley & Sons, New-York, pp. 1143-1169.

Lebart, L., 1969. Analyse statistique de la contiguïté. Publication de l'Institut de Statistiques de l'Université de Paris, 28: 81-112.

Levin, S.A., 1992. The problem of pattern and scale in ecology. Ecology, 73: 1943-1967.

Manly, B.F., 1994. Multivariate Statistical Methods. Chapman & Hall, London.

Misiti, M., Misiti, Y., Oppenheim, G., & Poggi, J.M., 1993. Ondelettes en statistique et traitement du signal. Revue de Statistique Appliquée, XLI: 33-43.

Pachepsky, Y.A., Ritchie, J.C., & Gimenez, D., 1997. Fractal modelling of airborne altimeter laser altimetry data. Remote Sensing of Environment, 61: 150-161.

Pachepsky, Y.A. & Ritchie, J.C., 1998. Seasonal changes in fractal landscape surface roughness estimated from airborne laser altimetry data. International Journal of Remote Sensing, 19: 2509-2516.

Parker, G.G., Lefsky, M.A., & Harding, D.J., 2001. Light transmittance in forest canopies determined using airborne laser altimetry and in-canopy quantum measurements. Remote Sensing of Environment, 76: 298-309.

Pitman, N.C.A., Terborgh, J., Silman, M.R., & Nuñez, P., 1999. Tree species distributions in an upper Amazonian Forest. Ecology, 80: 2651-2666.

Rango, A., Chopping, M., Ritchie, J.C., Havstad, K., Kustas, W., & Schmugge, T., 2000. Morphological characteristics of shrub coppice dunes in desert grasslands of southern New Mexico derived from scanning LIDAR. Remote Sensing of Environment, 74: 26-44.

Scheidegger, A.E., 1991. Theoretical Geomorphology. Springer Verlag, New York.

Strang, G. & Nguyen, T., 1996. Wavelets and filter banks. Wellesley-Cambridge Press, Wellesley, MA.

Thioulouse, J., Chessel, D., Dolédec, S., & Olivier, J.M., 1997. ADE-4: a multivariate analysis and graphical display software. Statistics and Computing, 7: 75-83.

Ver_Hoef, J.M., Cressie, N.A.C., & Glenn-Lewin, D.C., 1993. Spatial models for spatial statistics: some unification. Journal of Vegetation Science, 4: 441-452.

Weishampel, J.F., Blair, J.B., Knox, R.G., Dubayah, R., & Clark, D.B., 2000. Volumetric lidar return patterns from an old-growth tropical rainforest canopy. International Journal of Remote Sensing, 21: 409-415.

Table1

Contingency table expressing the cross-classification of the 257 transects into the three landscape units and the four clusters of patterns (row percent).

	cluster 1	cluster 2	cluster 3	cluster 4	TOTAL
A: alluvial plain (altitude<20m)	55	10	14	21	108
B: relief with complex forms (altitude<60m)	23	16	31	30	105
C: relief with complex forms (60m>altitude)	9	52	23	16	44
TOTAL	88	51	58	60	257

Figures

Figure 1

Example of reference patterns, used for standardization, for a transect length of n=64 and scales $b = \{1, 2, 4, 8, 16, 24\}$. Each of these patterns is deduced from the first eigenvector of the corresponding matrix $\mathbf{A_b}$.

Figure 2

Maps of general situation and sampling design of the Counami study site. (a) General situation. (b) Location of sampled transects (represented by the position of their midpoint). (c) Example of an unitary laser profile. (d) Maps of the three landscape units (A,B,C): for each unit, the texture of the relief is made apparent via an excerpt of a radar image, while the rest of the map is left blank.

Figure 3

Ordination and classification of the 257 scalograms. (a) Histogram of eigenvalues. (b) Biplot based on axes 1 and 2 of the principal components analysis (PCA) of the scalograms table $\mathbf{Y}$. The biplot technique allows a simultaneous plotting of individuals (transects, denoted by points) and variables (block sizes, represented by grey labels). The identifying number of some transects, evoked in the text, is mentioned on the plot (white labels). Straight lines are linking transects to the gravity center of the cluster to which they belong (clusters are labelled from 1 to 4). Ellipses are based on the clusters' standard deviations.

Figure 4

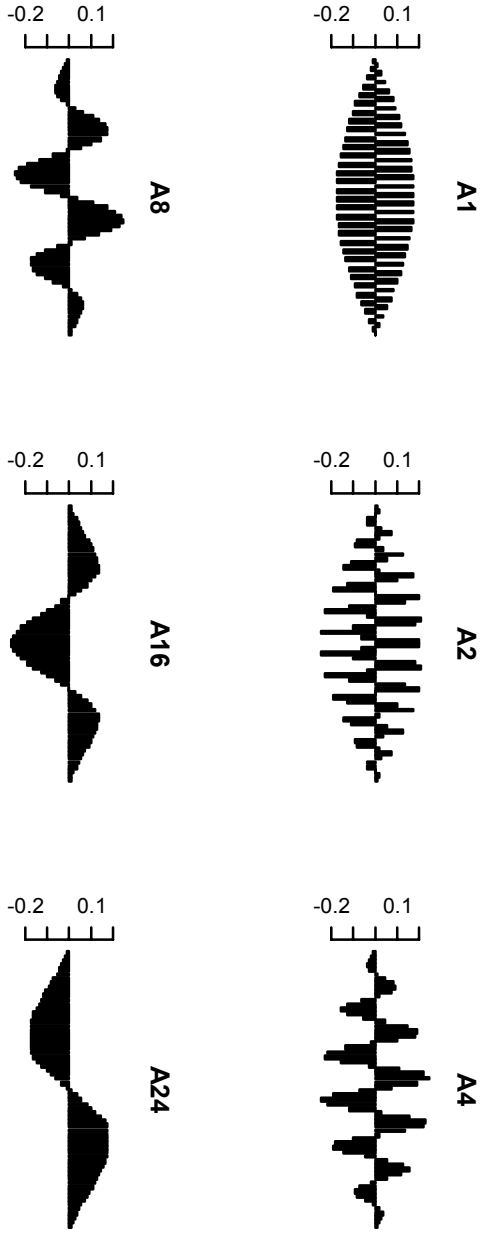
Dendrogram yielded by the hierarchical cluster analysis of the scalograms table $\mathbf{Y}$.

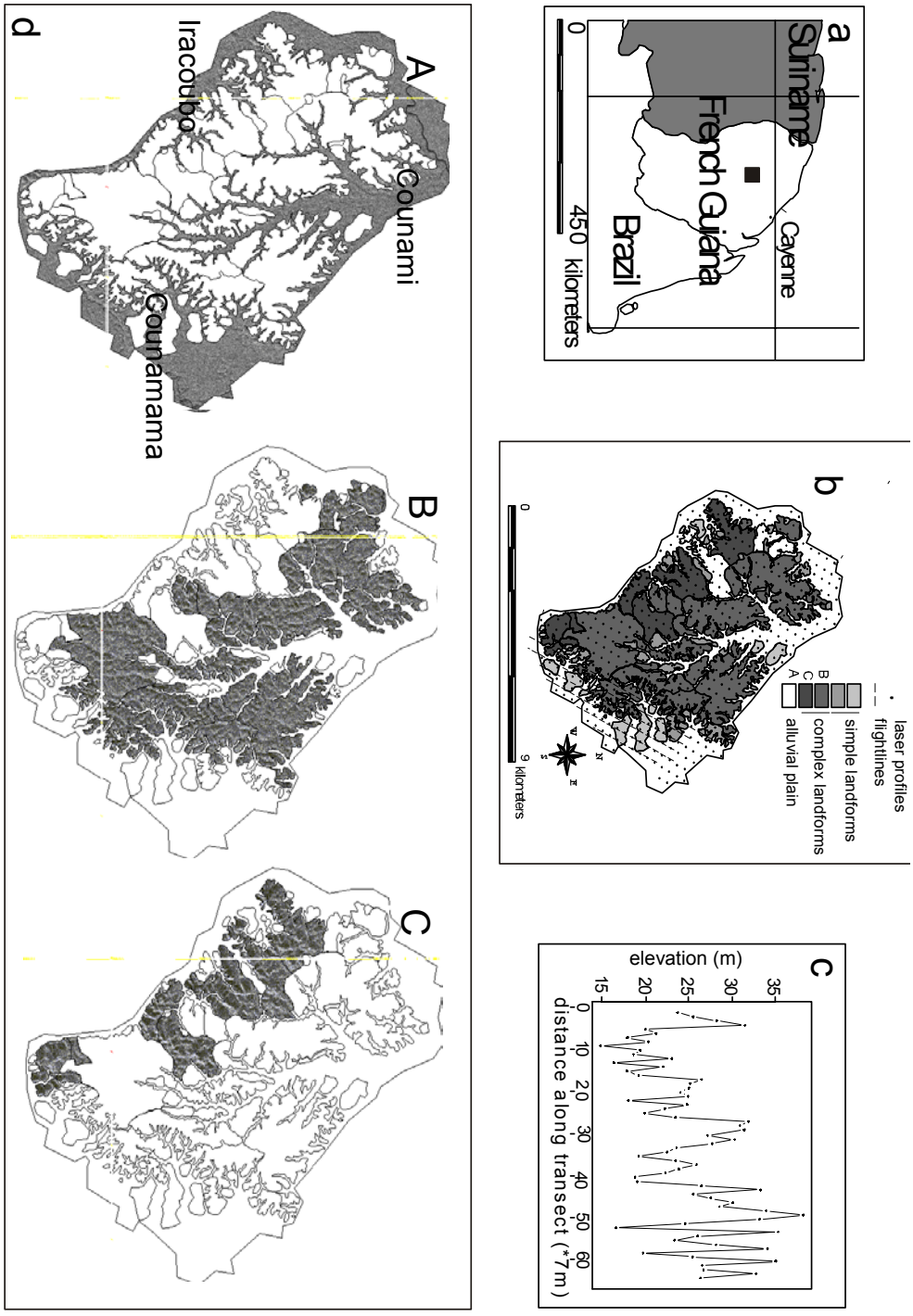
Figure 5

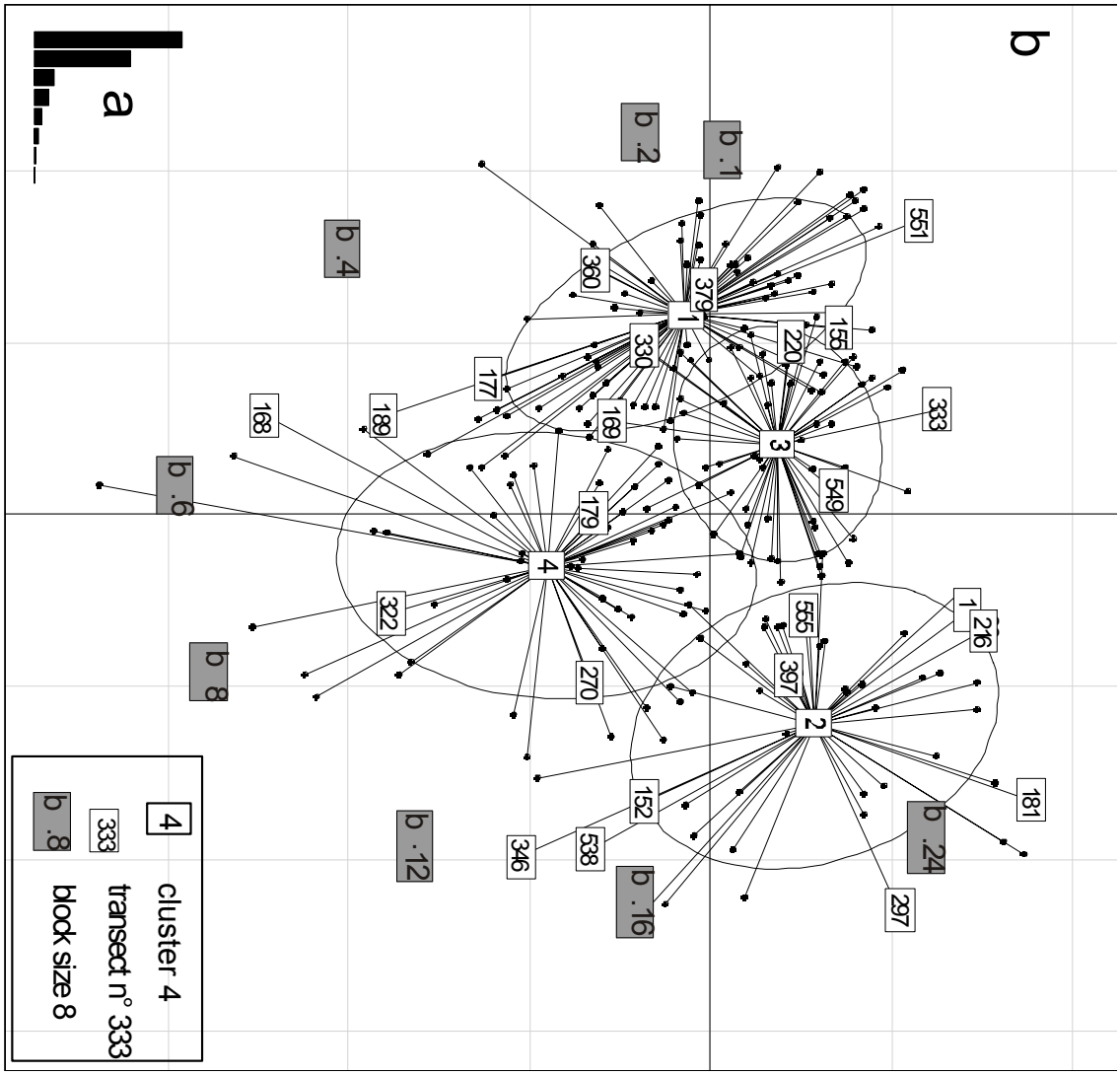
Average scalograms and typical laser profiles for clusters (a) and landscape units (b). The raw laser profile of the transect showing the smallest Euclidean distance to the average scalogram is displayed for each cluster and each landscape unit.

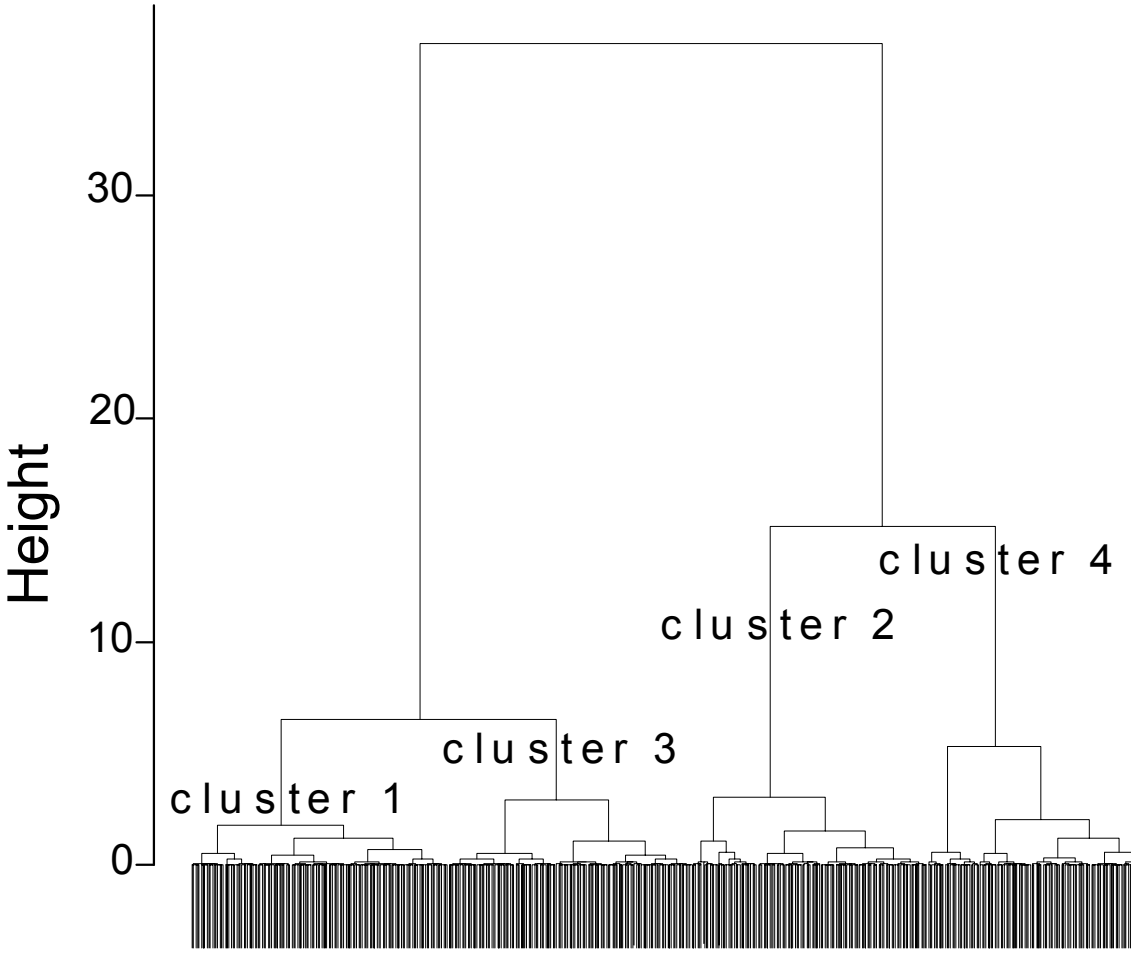
Figure 6

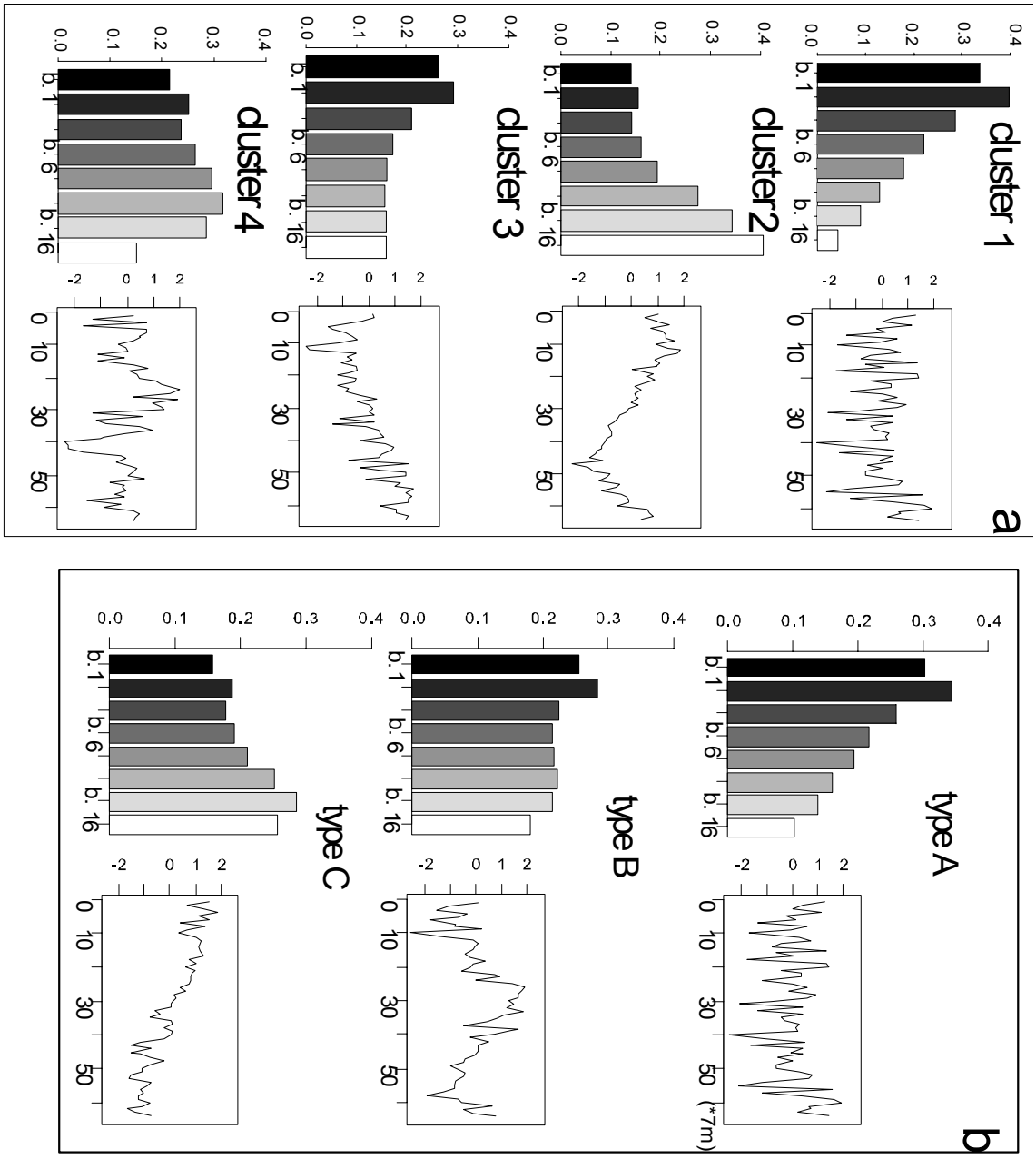
Spatial relationship between clusters and landscape units. Transects belonging to the focal cluster are represented by their midpoint.

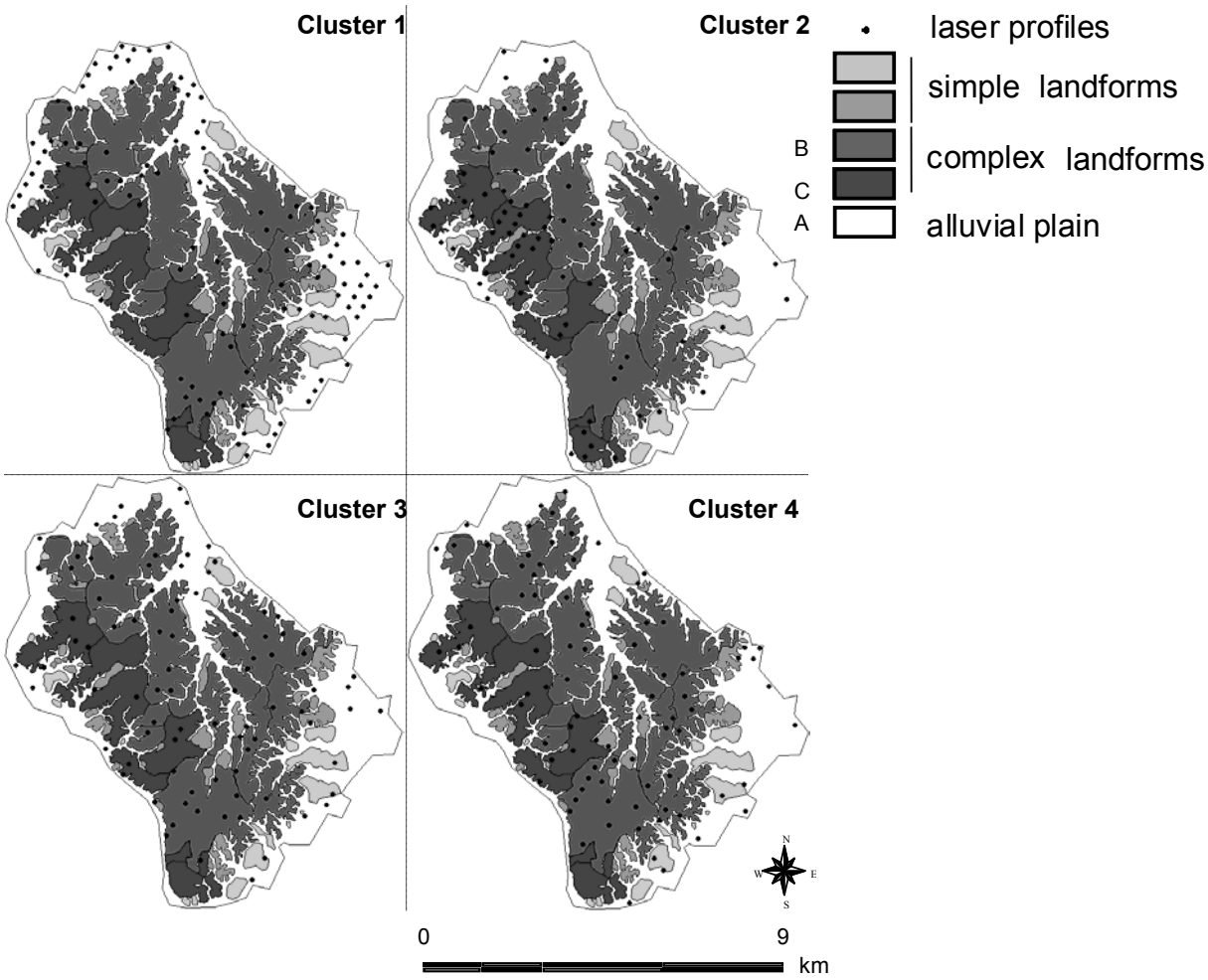

-0.2
0.1
A8
A1
A16
A2
A24
A4


a
Suriname
French Guiana
Brazil
Cayenne
0
450 kilometers
b
laser profiles
flightlines
simple landforms
complex landforms
alluvial plain
A
B
C
0
9 kilometers
c
elevation (m)
15
20
25
30
35
0
10
20
30
40
50
60
distance along transect (*7m)
d
A
Counami
Iracoubo
Counamama
B
C


a
b
b .1
b .2
b .4
b .6
b .8
b .12
b .16
b .24
1
2
3
4
551
360
379
155
220
330
177
168
189
169
333
549
179
322
555
1
216
397
270
152
181
538
346
297
4
cluster 4
333
transect n° 333
b .8
block size 8


Height
30
20
10
0
cluster 1
cluster 3
cluster 2
cluster 4


cluster 1
cluster2
cluster 3
cluster 4
a
type A
type B
type C
b
b. 1
b. 6
b. 16
0.0
0.1
0.2
0.3
0.4
-2
0
1
2
0
10
30
50
(*7m)


Cluster 1
Cluster 2
Cluster 3
Cluster 4
laser profiles
simple landforms
complex landforms
alluvial plain
B
C
A
N
W
E
S
0
9
km

4. Orthonormal transform to decompose the variance of a life-history trait across a phylogenetic tree

auteurs

Sébastien ollier
Pierre Couteron
Daniel Chessel

résumé

In recent years, there has been an increased interest in studying the variability of a quantitative life history trait across a set of species sharing a common phylogeny. However, such studies have suffered from an insufficient development of statistical methods aiming at investigating into phylogenetic dependence, i.e., the potential correlation between trait values due to the topological structure of the phylogenetic tree. We propose, here, a new approach that expresses the topological properties of the phylogenetic tree via an orthonormal basis, which is further used to decompose the trait variance. Such a decomposition provides a structure function, referred to as 'orthogram', which is relevant to characterise the phylogenetic dependence in both graphical and statistical aspects. We also propose four complementary test statistics to be computed from orthogram values, that help to diagnose both the intensity and the nature of phylogenetic dependence. The relevance of the method is illustrated by the analysis of three phylogenetic data sets, drawn from the literature and typifying contrasted levels and aspects of phylogenetic dependence. Freely available routines which have been programmed in the R framework are also proposed.

mots-clés

Phylogenetic tree, orthonormal transform, variance decomposition, life history trait, orthogram

journal

Biometrics: accepté, sous presse

Introduction

In recent years, there has been considerable interest in analysing the variability of one or several life-history traits, expressed as quantitative variables, across a set of species sharing a common phylogeny. Such studies find their justification in the "comparative method" (Harvey and Pagel, 1991), which provides a renewed and enlarged vision of an old and central paradigm of natural sciences. Comparative studies have investigated the way in which two life-history traits may correlate across a phylogenetic tree, as well as the correlation between one trait and an environmental variable. A serious statistical problem, affecting all these studies, arises from the fact that species which are part of a hierarchically structured phylogeny cannot be a priori considered as independent observations of a life trait variable (Felsenstein, 1984). Very often the presence of “phylogenetic signal” (Blomberg, Garland, and Ives, 2003) also called “phylogenetic autocorrelation” (Gittleman and Kot, 1990) is to be expected which means, for instance, that the difference in trait values observed for a given pair of species is likely to depend on their relative position in the phylogenetic tree.

The potential dependence of the values taken by the trait in relation to the topological structure of the phylogenetic tree, i.e., “phylogenetic dependence”, deserves great interest, not only because it may violate the main assumptions on which current statistical tests are built, but also because a proper characterisation may provide valuable insights on the processes and mechanisms that have shaped the data under consideration. This has long been acknowledged (Gittleman and Kot, 1990) in a trend analogous to the increased interest for the study of temporal and/or spatial dependence that has been observed in ecological sciences (Legendre, 1993).

However, concepts and methods originating from time series analysis and spatial statistics proved to be of a more direct applicability to ecological data than to phylogenetic topics. Central to these methods are "structure functions" (Legendre and Legendre, 1998), such as correlogram, variogram, periodogram (Ripley, 1981; Cressie, 1991; Wackernagel, 2003) or wavelet-based “scalograms” (Percival and Walden, 2000) which are useful to describe, model and test spatial or temporal patterns. Graphic tools (e.g., more or less sophisticated geographical maps) are also widely used as complements of the summary provided by structure functions. In contrast to this well-established, yet still burgeoning, field developments addressing the description of phylogenetic dependence have remained limited. Many tests dealing with phylogenetic issues have been proposed (Blomberg et al., 2003), several of them addressing the significance of the correlation between a trait variable and the

phylogenetic tree. But there is no method providing a satisfactory description of how the variance of a quantitative variable is distributed along a phylogenetic tree. In particular, the application of nested analysis of variance (Bell, 1989) cannot completely accommodate full phylogenetic information (Blomberg et al., 2003), while the only published structure function is an extension to phylogeny of the correlogram concept (Gittleman and Kot, 1990), which is far from having optimal properties (Rohlf, 2001), just as the usual correlogram is generally not the most relevant structure function to address spatial or temporal dependence (Ripley, 1981). In addition, the lack of graphical standards is blatant in the literature devoted to phylogeny, in which simultaneous displays of trait values and of the reference phylogenic tree are very scarce.

The main object of this paper is to provide a canonical procedure to decompose the variance of a life-trait variable with respect to the topological structure of a phylogenetic tree, and to propose a new structure function, called an "orthogram", to express such a decomposition on an orthonormal basis constructed from the topology of the tree. Additionally, we intend to provide some complementary non parametric test statistics derived from the orthogram, along with principles for a simultaneous graphical display of both trait values and phylogeny. The effectiveness of the approach will be illustrated and compared with the phylogenetic correlogram, using examples drawn from the literature and showing contrasted patterns of phylogenetic dependence.

An orthonormal basis to decompose trait variance

Building an orthonormal basis from the topological structure

Let us consider a life history trait represented by a quantitative variable $\mathbf{x} = \left(x_1, ..., x_t\right)^T$ measured on t taxonomic units, for which the topology of the phylogenetic tree is assumed to be a priori fully determined. We shall consider the following terminology concerning the phylogenetic tree. The root is the common ancestor to all the t contemporary taxons (OTUs: operational taxonomic units, also called "tips") and to the n hypothetical nodes (HTUs: hypothetical taxonomic units); branches emanate from root and nodes, tracing the course of evolution. Phylogenetic trees used by biologists are generally "consensus trees" (Adams, 1972) or "super-trees" (Sanderson, Purvis and Henze, 1998) obtained by compiling phylogenies established by distinct methods, which means that branch lengths are often unknown or not accurate.

Even when data are from homogeneous origins, the estimation of distances between tips from either fossil, morphological or molecular data relies on the hypothesis that all traits evolve at a steady rate along all branches. But evolutionary rates are known to change with time (Svensson, 1997) while varying between branches of a phylogenetic tree (Mindell and Thacker, 1996) and this often introduces a substantial level of uncertainty around estimates of branch lengths. Consequently, we shall restrict ourselves to the most general type of data for which only the topological structure of the phylogenetic tree, i.e., the relationships between nodes and between nodes and tips, can be assumed to be known.

Each taxonomic unit of the tree (either tip or node) defines a set made of its descendant tips. Thus, we can associate a dummy variable $\mathbf{u}^k = \left[u_i\right]_{1\le i\le t}^k$ to each taxonomic unit *k* (including the root, *1*<*k*<*n*+*t*+*1*), with $u_i^k = 1$ if the tip i is descended from that taxonomic unit *k*, and $u_i^k = 0$ otherwise. For example, the node N2 in figure 1A defines a subset of tips which is $\{\text{L7,L8,L9,L10,L11,L12}\}$, while the corresponding dummy variable is $\mathbf{u}^T = (0,0,0,0,0,0,1,1,1,1,1,1)$. For the subsequent analyses, we need to define a relative order between dummy variables and, thus, between the corresponding taxonomic units (TU). A reasonable quantitative criterion to rank the TU is the number, $np(k)$, of permutations between descendants of a given node *k* which preserve the topological structure of the sub-tree stemming from it. With such a criterion, nodes are ranked according to the complexity of the sub-tree they initiate. Let us now consider the table $\mathbf{U} = \left[\mathbf{u}^k\right]$ that contains the whole sets of the dummy variables.

All these dummy variables are obviously not linearly independent. The most obvious reason is that the dummy variables corresponding to the direct descendants of a given node, *k*, sums to the dummy variable, $\mathbf{u}^k$. This problem can be easily resolved by suppressing for each node, *k*, the dummy variable corresponding to its direct descendant displaying the lowest np value. This means eliminating $n+1$ dummy variables (the root is considered as a node) among the $n+t+1$ possible, i.e., retaining only *t* of them. The next step is a row-wise concatenation of these *t* dummy variables to form a *t x t* matrix, **V** (figure 1A). The QR decomposition (Harville, 1997) of matrix **V** yields a *t x t* matrix, **Q**, whose columns are the orthogonal vectors obtained by applying Gram-Schmidt orthogonalization to the columns of **V**.

The orthonormal basis **B** associated with the phylogeny is defined from the orthogonal basis **Q**. We eliminate the first vector $\mathbf{1}_t$ corresponding to the dummy variable attached to the root

and we standardize by $\sqrt{t}$ to obtain $t-1$ orthonormal vectors (figure 1B). Orthonormality guaranties that $\frac{1}{t}\mathbf{B}^T\mathbf{B} = \mathbf{I}_{t-1}$. The vectors of **B** are linear combinations of the dummy variables and the Gram-Schmidt orthogonalization guarantees that each of these vectors is ranked in accordance with the initial ranking of the dummy variables. Consequently, the first vectors will characterize interspecific variance among dissimilar species (two species are all the more dissimilar that the *np* value of their first common ancestor is high) and reciprocally. This enables an interpretation of the successive vectors in terms of decreasing phylogenetic dissimilarity between tips.

######### insert figure 1 #########

Decomposition and reconstruction of the life trait variable

From the orthonormal basis **B**, we can compute a vector $\mathbf{r} = (r_1,...,r_{t-1})^T$ of the transform coefficients of the centred and standardized variable $\mathbf{x_0}$, as $\mathbf{r} = \frac{1}{t}\mathbf{B}^T\mathbf{x_0}$. Centring and standardizing are carried out using the uniform weighting of tips and, thus, the usual definitions for mean and variance. This is a purely technical choice that allows us to have cumulated orthogram values between 0 and 1 thereby facilitating comparisons between examples.

Transform coefficients allow one to reconstruct the variable using $\mathbf{x_0} = \mathbf{Br}$, while squared transform coefficients provide a decomposition of its variance : $\|\mathbf{x_0}\| = \frac{1}{t}\mathbf{x_0}^T\mathbf{x_0} = \frac{1}{t}\mathbf{r}^T\mathbf{B}^T\mathbf{Br} = \mathbf{r}^T\mathbf{r}$. Squared coefficients, i.e. $\left(r_i^2\right)_{1\le i\le t-1}$, as well as cumulative squared coefficients, can be plotted against $i \in \{1,...,t-1\}$ yielding two graphical tools, that we shall call orthogram and cumulative orthogram, respectively. The sign of coefficients can also be mentioned for orthogram values (see figures 2 to 4). Both types of orthograms provide a means to display the variance decomposition for the trait while enabling several tests for phylogenetic dependence. As we shall see in the examples, the ordering of orthonormal vectors by decreasing *np* values of the corresponding dummy variables enhances the interpretability of orthogram results

Testing for phylogenetic dependence

We consider as null hypothesis (H_0) the complete absence of phylogenetic dependence (Bloomberg et al., 2003) and, thus, that the *t* observed tip values are exchangeable irrespective of the topological structure of the tree. This means that the observed decomposition of the trait variance is to be compared with decompositions obtained from a sample of the *t*! permutations defined on the elements of $\mathbf{x} = [x_i]_{1 \le i \le t}$, and for which the orthogram is expected to have uniform values. To carry out such a comparison, we introduced four statistics computed from the variance decomposition on which tests of the null hypothesis are based. The four tests are complementary in the sense that they may have different relevance and power in the presence of distinct alternatives to H_0 (see below).

A first test is built on the statistic,

$$\mathrm{R2Max}(\mathbf{x}) = \max\left(r_1^2, ..., r_{t-1}^2\right)$$

which is expected to peak when a unique vector of the basis accounts for a large share of the trait variance. This would mean that a significant change in the life trait appeared at one node of the phylogenetic tree while being conserved in the deriving branches.. This test is likely to be of limited relevance in the presence of a diffuse dependence, *i.e.,* when several nodes are prominent to explain the trait variance and when the orthogram is therefore dominated by several large values instead of a unique sharp peak. In an analogy with what has been proposed to test smooth periodograms (Bartlett, 1954), we derived a second statistic from the cumulative orthogram, namely:

$$D\max(\mathbf{x}) = \max_{1 \le m \le t-1}\left(\sum_{i=1}^{m} r_i^2 - \frac{m}{t-1}\right).$$

It corresponds exactly to the Kolmogorov-Smirnov statistic used to test whether the vector $\left[\sum_{i=1}^{m} r_i^2\right]_{1 \le m \le t-1}$ may be an ordered random sample from the uniform distribution on $(0,1)$. The third test is built on the statistic:

$$\mathrm{SkR2k}(\mathbf{x}) = \sum_{i=1}^{t-1} i r_i^2$$

that assesses to what extent the variance distribution across the phylogenetic tree is rather skewed to the root or to the tips. The last test corresponds to the statistic:

$$\mathrm{SCE}(\mathbf{x}) = \sum_{i=1}^{t-1}\left(\mathrm{r}_i^2 - \mathrm{r}_{i-1}^2\right)^2$$

which measures the average local variation of the orthogram values.

For all four statistics, confidence envelopes are built from a relevant number of Monte-Carlo randomizations of the x_i values. All computations and graphical displays involved in the preparation of this paper were carried out using R (Ihaka and Gentleman, 1996), with both pre-programmed and personal routines. These routines will be incorporated in the ade4 package available at http://lib.stat.cmu.edu/R/CRAN/, and in the meantime are available by a simple request to the first author.

Applications

Phylogenetic examples came from several papers that studied the degree to which phylogenetic history has shaped the evolution of phenotypic characters or life traits. We retained three examples that typify contrasted levels and aspects of phylogenetic dependence. For each example we compared the results provided by the orthogram with the results yielded by the only other proposed structure function, namely the phylogenetic correlogram (Gittleman and Kot 1990). The correlogram is constructed by computing Moran's I coefficients (Cliff and Ord, 1981), for successive classes of phylogenetic distance. Detailed procedures for computing correlogram values and their corresponding confidence intervals are provided by Gittleman and Kot (1990).

Exploratory analyses of life trait evolution should start with meaningful displays of the variation of the focal trait along the phylogenetic tree, just as most quantitative studies of spatially structured phenomena usually start with a visual analysis using geographical maps of the variables under study. But such displays are still greatly lacking in phylogenetic literature and we have therefore proposed a simple graphical method. We represent the tree in front of the dotplot of the variable (figure 2A). This graphical representation takes a leaf out of Cleveland's book (1994) for it was he who defined the dotplot '*as a graphical method for measurements that have labels*'. As for geographical maps in the spatial domain, this graphical representation will help to give a first idea about the nature of the patterns, while enhancing the interpretation of quantitative results.

a. Absence of phylogenetic dependence

We first analysed data provided by Bried, Pontier, and Jouventin (2002), about the adult life expectancy of 18 species of Procellariiformes (figure 2A). Phylogenetic relationships were compiled from various information sources available for procellariiformes (see Bried, Pontier and Jouventin, 2002 for details). Because divergence times between species are far from certain (Pontier, personal communication), lengths of branches are considered as unknown.

The trait variable of life expectancy has been square-root transformed prior to analysis so as to make its distribution more symmetric and normal. None of our four tests rejected the null hypothesis of a uniform distribution of orthogram values, since the observed values of the four corresponding statistics were all exceeded by results of many Monte-Carlo randomizations (histograms in figure 2B). Furthermore, values of the cumulated orthogram remained within the confidence envelopes (figure 2B). All these results pointed to an absence of phylogenetic dependence for that life history trait. The phylogenetic correlogram of observed data (figure 2C) also indicated an absence of phylogenetic dependence whatever the distance class. Thus, this example supports the idea that not all traits are correlated to their phylogenetic history. Indeed, there exist specific evolutionary scenarios (Blomberg et al., 2003) under which correlation between a trait and its phylogenetic history is likely to be low. Although testing for the absence of phylogenetic dependence could appear trivial, it is a first and unavoidable step to study the relationship between a life trait and a phylogeny, as shown by this example.

######### insert figure 2 #########

b. Importance of a particular node

For this example, we analysed a data set from Pélabon et al. (1995) relating to the adult female body weight of 18 species of Ungulates, using their tip data and the joint phylogeny (figure 3A). Because the phylogenetic relationships are unknown for ungulates (Gaillard, personal communication), a taxonomy-based phylogeny with unknown branch lengths has been used. The trait has been log-transformed prior to analysis. Only the test based on R2Max was significant (first histogram in figure 3 B). Moreover, the plot of orthogram values highlighted the prevalence of the fourth vector of the basis, which corresponds to the node 'w10' (figure 3 B). The simultaneous representation of that score and trait values (figure 3A) clearly confirmed the presence of an important evolutionary event that appeared at node w10 and conserved until present. Such an evolutionary pattern generates a kind of phylogenetic dependence which appeared hard to detect by three of the test statistics (SkR2k, DMax, SCE), by the cumulative orthogram and also by the phylogenetic correlogram (figure 3C). Only the multiple tests on individual values of the orthogram and the test statistic R2Max proved able to indicate a significant departure from phylogenetic independence.

######### insert figure 3 #########

c. Diffuse phylogenetic dependence

The last example analysed deals with the age at maturity (years) of 49 species of teleost fishes (figure 4A). Data have been provided by Rochet et al. (2000) who compiled the most

recent information available for teleost fishes to establish phylogenetic relationships. Partial phylogenetic trees based on morpho-anatomical characters and on molecular traits were collated to yield a consensus tree summarising present knowledge about teleost interrelationships. As data were obtained from different sources and methods, estimates of branch lengths are not available or comparable. The life trait was log-transformed prior to the analysis. This trait can be considered as shaped by phylogenetic history since three test statistics (SkR2k, DMax, SCE) revealed a significant departure from H_0 while the cumulative orthogram had most of its values outside the confidence envelopes (figures 4B). This pattern of phylogenetic dependence is radically different from the preceeding one. In this example, the values of the orthogram and, thus, the portions of interspecific variance, decreased regularly as a function of the complexity value, np, of the nodes. The phylogenetic correlogram (figure 4C) confirmed the existence of a "phylogenetic gradient": there was a monotonic decrease of coefficients in relation to phylogenetic distance. This profile indicated that closely related species tend to have similar trait values and that such a similarity decreases with phylogenetic distance.

######### insert figure 4 #########

Discussion

The above examples demonstrate that orthonormal transform is a relevant approach to diagnose different degrees and types of phylogenetic dependence in both small and large phylogenies, as well as with different types of phenotypic characters. The approach is all the more relevant in that modern computer technology with its high-power graphic screens displaying multiple, linkable windows allows one to consider dynamic simultaneous views on the phylogenetic data and on the distributions of the values taken by the structure functions and test statistics.

Characterising phylogenetic dependence using an orthonormal transform comprises two main steps, namely (i) the definition of an orthonormal basis $\mathbf{B}$, which ensures a canonical description of the tree topology and (ii) the variance decomposition on the basis that yields the *structure functions* called orthograms. Such a strategy is generic in the sense that it does not necessitate any assumption on the structure of the tree and therefore contrasts with several *ad-hoc* methods which have been previously proposed. Furthermore, the principle of orthonormal transforms obviously surpasses the study of phylogenetic data. Indeed, in the spatial and/or temporal domain, the Fourier transform as well as several kinds of wavelet

transforms (Percival, 2002) also rely on specific orthonormal bases (Percival and Walden, 2000). Furthermore, all the canonical bases associated with linear and bidimensional supports such as the eigenvectors of a graph matrix (Cvetkoviv, Doob, and Sachs, 1979) could be used to calculate spatial and temporal orthograms. Thus, the eigenvectors of a matrix of inter-tips phylogenetic distances (sensu Rohlf, 2001) can provide an alternative way to compute a phylogenetic orthogram.

Consequently, the four tests statistics we have used can also be applied to structure functions which are analogues of the orthogram. Statistics (R2Max) and (Dmax) have long been proposed for the Fourier periodogram (Bartlett, 1954; Diggle, 1990), but they are obviously relevant in regards to wavelet decompositions using an orthonormal basis. This is also true with respect to the third and the fourth statistics whose use in relation to structure functions is new to our knowledge. There are, indeed, many extensions of this work which should be examined in the future. As with so many tests aimed at pattern detection, no information on power is yet available. The three examples we did provide as illustrations cannot claim to encompass all the diversity of phylogenetic patterns, though they clearly illustrate that the different statistics and structure functions are likely to be of varying pertinence and power when facing contrasted alternatives to phylogenetic independence (see in figure 3 the limited power of both phylogenetic correlogram and cumulative orthogram with respect to the second example). The orthogram provides a very rich account of variance decomposition since there are as many values as tips (minus 1) in the tree, while the correlogram only gives averaged results for a limited number of classes of phylogenetic distances. This is analogous to the limitation encountered in the temporal/spatial domain with the correlogram (or the closely related Moran's Index, Cliff and Ord, 1981), when compared with a wavelet decomposition which can also provide results for individual observations.
In the future, the different properties of structure functions and test statistics could be useful to identify which kind of phylogenetic dependence is present (e.g., diffuse vs. structured by particular nodes) by considering which particular tests reject the null hypothesis and those which do not. However, in order to do so on a properly established basis it is still necessary to study and assess the power of these tests with regards to phylogenetic data, simulated under various models of evolutionary change, and having contrasted characteristics in terms of branch lengths and divergence time. Moreover, the ultimate goal of most comparative studies being the analysis of the evolutionary patterns of several life traits (Felsenstein 1984), a

multivariate extension of our orthonormal basis approach is, obviously, the next step to consider in the near future.

Acknowledgements

We are grateful to D. Pontier and J.M. Gaillard who have placed their comparative data sets at our disposal in order to evaluate the relevance of our method. We are indebted to two anonymous referees for important methodological comments that greatly helped us enhance the initial version.

References

Adams, E.N. (1972). Consensus techniques and the comparison of taxonomic trees. Systematic Zoology 21, 390-397.

Bartlett, M. S. (1954). Problèmes de l'analyse spectrale des séries temporelles stationnaires. Publications de l'Institut de Statistique de l'Université de Paris 2, 119-134.

Bell, G. (1989). A comparative method. American Naturalist 149, 91-111.

Blomberg, S. P., Garland, T. and Ives, A.R. (2003). Testing for phylogenetic signal in comparative data. Evolution 57, 717-745.

Bried, J., Pontier, D., and Jouventin, P. (2003). Mate fidelity in monogamus birds: a re-examination of the Procellariiformes. Animal Behavior 65, 235-256.

Cleveland, W. S. (1994). The elements of graphing data. New Jersey: AT&T Bell Laboratories, Murray Hill.

Cliff, A.D. and Ord, J.K. (1981). Spatial Process. Pion.

Cressie, N. A. C. (1991). Statistics for Spatial Data. New York: Wiley & Sons.

Cvetkoviv, D. M., Doob, M. and Sachs, H. (1979). Spectra of graphs. New York: Academic Press.

Diggle, P. J. (1990). Time Series: a biostatistical introduction. Oxford: Clarendron Press.

Felsenstein, J. (1985). Phylogenies and the comparative method. The American Naturalist 125, 1-15.

Gittleman, J. L. and Kot, M. (1990). Adaptation: statistics and a null model for estimating phylogenetic effects. Systematic Zoology 39, 227-241.

Harvey, P. H. and Pagel, M. (1991). The Comparative Method in Evolutionary Biology. Oxford: Oxford University Press.

Harville, D. A. (1997). Matrix algebra from a statistician's perspective. New York: Springer.

Ihaka, R. and R. Gentleman (1996). R: a language for data analysis and graphics. Journal of Computational and Graphical Statistics 5, 299-314.

Legendre, P. (1993). Spatial autocorrelation: trouble or new paradigm. Ecology 74, 1659-1673.

Legendre, P. and Legendre, L. (1998). Numerical ecology. Amsterdam: Elsevier Science BV.

Mindell, D.P. and Thacker, C.E. (1996). Rates of molecular evolution: phylogenetics issues and applications. Annual Review of Ecology and Systematics 27, 279-303.

Pélabon, C., Gaillard, J. M., Loison, A., and Portier, C. (1995). Is sex-biased maternal care limited by total maternal expenditure in polygynous ungulates? Behavioral Ecology and Sociobiology 37, 311-319.

Percival, D. (2002). Wavelets. In Encyclopedia of Environmetrics, A. H. El-Shaarawi and Piegorsch, W. W. (ed), 4, 2338-2351. Chichester: John Wiley & Sons, Ltd.

Percival, D. B. and Walden, A. T. (2000). Wavelet Methods for Time Series Analysis. Cambridge: Cambridge University Press.

Ripley, B. D. (1981). Spatial Statistics. New York: John Wiley and Sons.

Rochet, M.J., Cornillon, P.A., Sabatier, R. and Pontier, D. (2000). Comparative analysis of phylogenetic and fishing effects in life patterns of teleost fishes. Oikos 91, 255-270.

Rohlf, F. J. (2001). Comparative methods for the analysis of continuous variables: geometric interpretations. Evolution 55, 2143-2160.

Sanderson, M.J., Purvis, A. and Henze, C. (1998). Phylogenetic supertrees: assembling the trees of life. Trends in Ecology and Evolution 13, 105-109.

Svensson, E. (1997). The speed of life-history evolution. Trends in Ecology and Evolution 12, 380-381.

Wackernagel, H. (2003). Multivariate geostatistics. An introduction with applications. Berlin: Springer.

Figures

Figure 1

Illustration of the orthonormal basis construction from a fictive phylogenetic tree featuring 12 tips and 7 nodes. A. Representation of *t-1* dummy variables, associated to nodes and tips, and ordered by decreasing values of *np* (names of the dummy variables are labelled on the top of the table). B. Representation of the *t-1* orthonormal vectors of the orthonormal basis **B**. For each node of the phylogeny, the labels indicate which vectors account for the variance associated to each node. The size of the squares is proportional to the values of the orthonormal vectors (white and black for negative and positive values, respectively).

Figure 2

Analysis of the adult life expectancy for 18 species of procelariiformes (Bried et al., 2003). A. Phylogenetic tree (on the left) with dotplot of the life expectancy variable (centre) and species names (on the right). B. Orthogram plot, cumulative orthogram plot and histograms of simulated values (999 Monte-Carlo permutations of tip values) with the position of the observed result for the four test statistics. Orthogram plot: the bars are proportional to the squared coefficients (white and grey bars stand for positive and negative coefficients, respectively). The dashed line is the upper confidence limit at 5%, deduced from 999 Monte-Carlo permutations (mean value indicated by the horizontal solid line). Cumulative orthogram plot: circles represent observed values of cumulated squared coefficients, expected values under H_0 are along the straight line and dashed lines stand for the bilateral confidence interval. C. Phylogenetic correlogram with six successive Moran's I coefficients (solid squares). The straight line indicates the expected value under H_0 and dashed lines the bilateral confidence interval at 5%, deduced from 999 Monte-Carlo permutations.

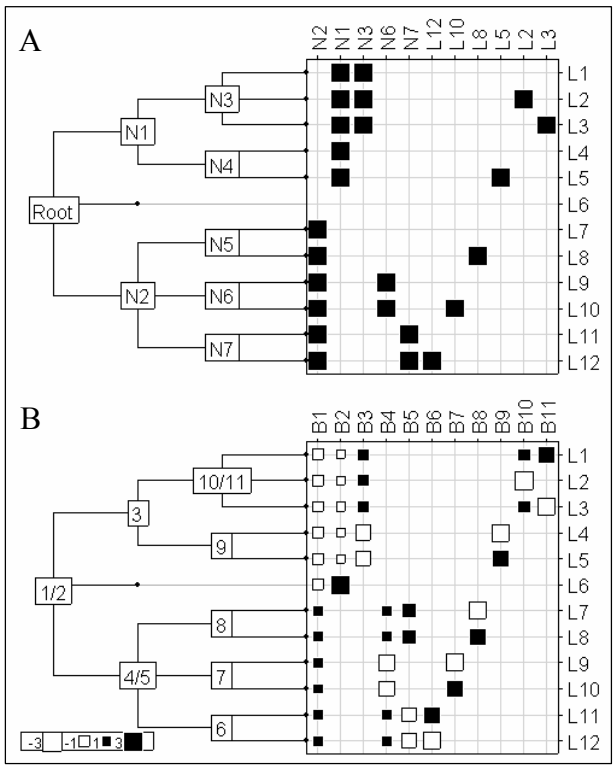
Figure 3

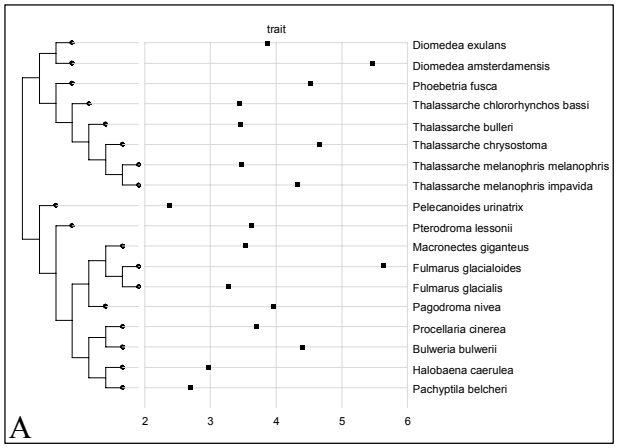
Analysis of the adult female body weight (afbw) of 18 species of Ungulates (Pélabon et al., 1995). Phylogenetic tree with labels of the nodes (on the left) and species names (on the right). At the centre, dot plot of the trait variable (afbw) and of the score attached to the fourth vector of **B**. B. Orthogram plot, cumulative orthogram plot and histograms of the simulated values (999 Monte-Carlo permutations of the tip values) with the position of the observed value for the four test statistics . Orthogram plot: the bars are proportional to the squared coefficients (white and grey bars stand for positive and negative coefficients, respectively). The dashed line is the upper confidence limit at 5%, deduced from 999 Monte-Carlo permutations (mean value indicated by the horizontal solid line). Cumulative orthogram plot:

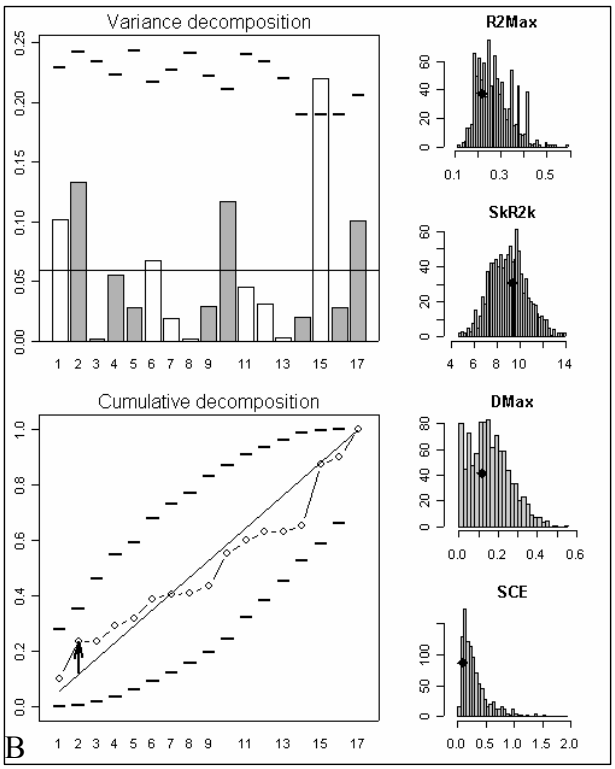
circles represent observed values of cumulated squared coefficients, expected values under H_0 are along the straight line and dashed lines stand for the bilateral confidence interval. C. Phylogenetic correlogram with five successive Moran's I coefficients (solid squares). The straight line indicates the expected value under H_0 and dashed lines the bilateral confidence interval at 5%, deduced from 999 Monte-Carlo permutations.

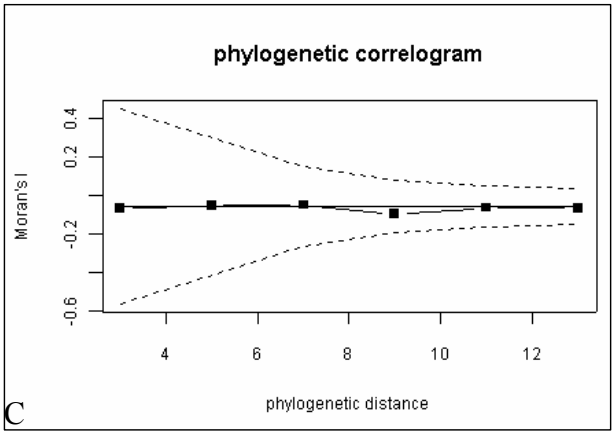
Figure 4

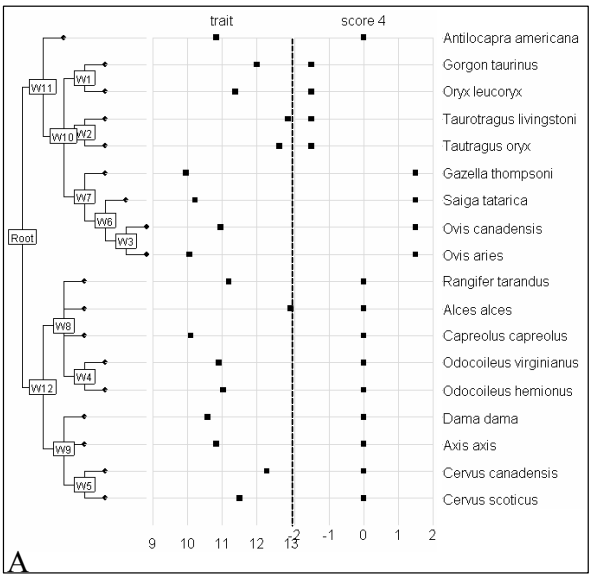
Analysis of the age at maturity (years) of 49 species of teleost fishes (Rochet et al., 2000). A. Phylogenetic tree (on the left) with dotplot of the age at maturity (centre) and species names (on the right). B. Orthogram plot, cumulative orthogram plot and histograms of simulated values (999 Monte-Carlo permutations of tip values) with the position of the observed result for the four test statistics considered. Orthogram plot: the bars are proportional to the squared coefficients (white and grey bars stand for positive and negative coefficients).The dashed line is the upper confidence limit at 5%, deduced from 999 Monte-Carlo permutations (mean value indicated by the horizontal solid line). Cumulative orthogram plot: circles represent observed values of cumulated squared coefficients, expected values under H_0 are along the straight line and dashed lines stand for the bilateral confidence interval. C. Phylogenetic correlogram with seven successive Moran's I coefficients (solid squares). The straight line indicates the expected value under H_0 and dashed lines the bilateral confidence interval at 5%, deduced from 999 Monte-Carlo permutations.

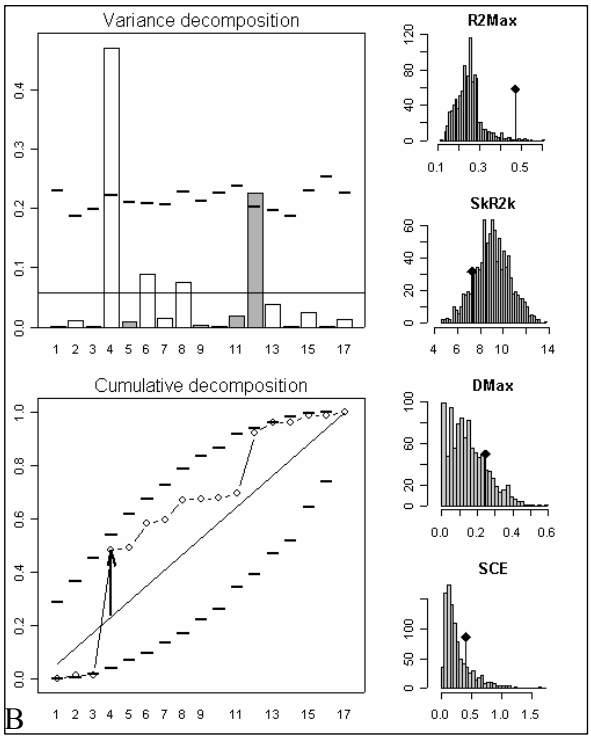

A
N2
N1
N3
N6
N7
L12
L10
L8
L5
L2
L3
L1
L2
L3
L4
L5
L6
L7
L8
L9
L10
L11
L12
N3
N1
N4
Root
N5
N2
N6
N7
B
B1
B2
B3
B4
B5
B6
B7
B8
B9
B10
B11
10/11
3
9
1/2
8
4/5
7
6
-3
-1
1
3

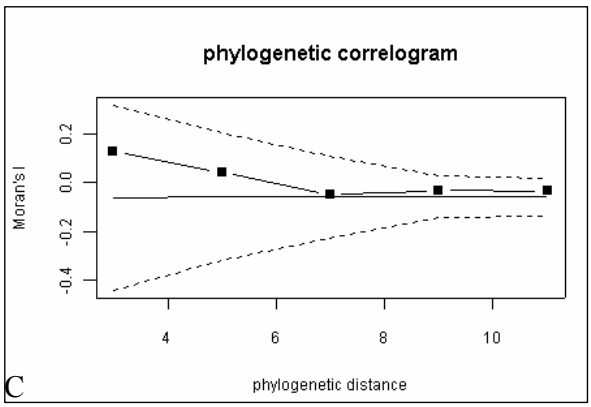

trait
Diomedea exulans
Diomedea amsterdamensis
Phoebetria fusca
Thalassarche chlororhynchos bassi
Thalassarche bulleri
Thalassarche chrysostoma
Thalassarche melanophris melanophris
Thalassarche melanophris impavida
Pelecanoides urinatrix
Pterodroma lessonii
Macronectes giganteus
Fulmarus glacialoides
Fulmarus glacialis
Pagodroma nivea
Procellaria cinerea
Bulweria bulwerii
Halobaena caerulea
Pachyptila belcheri
2 3 4 5 6
A

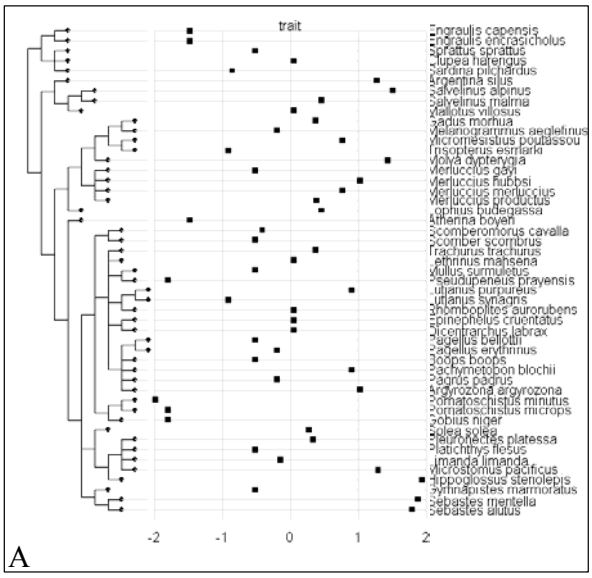

Variance decomposition
R2Max
SkR2k
Cumulative decomposition
DMax
SCE
B

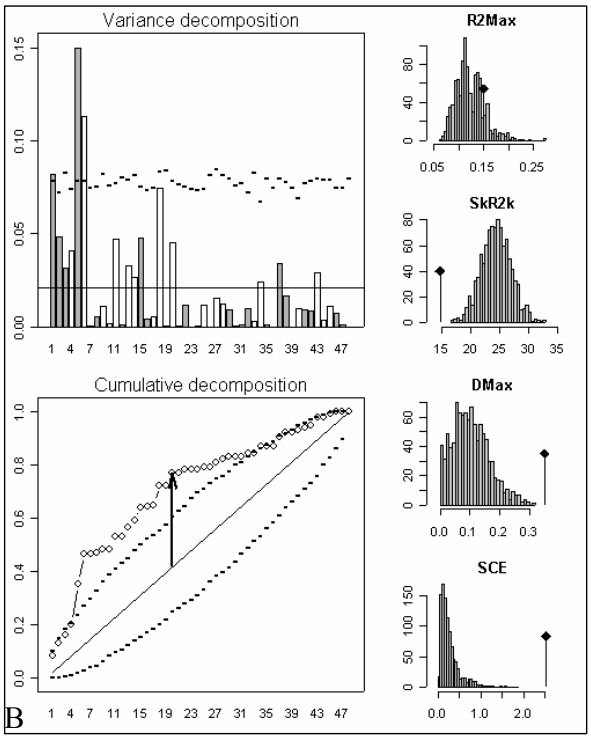

phylogenetic correlogram
Moran's I
phylogenetic distance
C

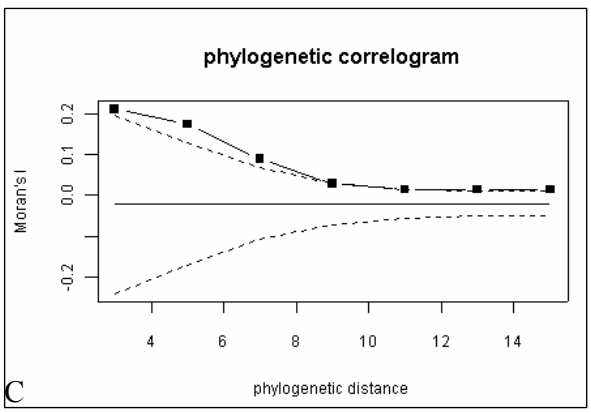

trait
score 4
Antilocapra americana
Gorgon taurinus
Oryx leucoryx
Taurotragus livingstoni
Tautragus oryx
Gazella thompsoni
Saiga tatarica
Ovis canadensis
Ovis aries
Rangifer tarandus
Alces alces
Capreolus capreolus
Odocoileus virginianus
Odocoileus hemionus
Dama dama
Axis axis
Cervus canadensis
Cervus scoticus
Root
W1
W11
W2
W10
W7
W6
W3
W8
W4
W12
W9
W5
9 10 11 12 13
-2 -1 0 1 2
A


Variance decomposition
Cumulative decomposition
R2Max
SkR2k
DMax
SCE
B


phylogenetic correlogram
Moran's I
phylogenetic distance
C


trait
Engraulis capensis
Engraulis encrasicholus
Sprattus sprattus
Clupea harengus
Sardina pilchardus
Argentina silus
Salvelinus alpinus
Salvelinus malma
Mallotus villosus
Gadus morhua
Melanogrammus aeglefinus
Micromesistius poutassou
Trisopterus esmarkii
Molva dypterygia
Merluccius gayi
Merluccius hubbsi
Merluccius merluccius
Merluccius productus
Ophius budegassa
Athenna boyeri
Scomberomorus cavalla
Scomber scombrus
Trachurus trachurus
Ethmnus mahsena
Mullus surmuletus
Pseudupeneus prayensis
Uranus purpureus
Uranus synagris
Rhombopltes auronubens
Epinephelus cruentatus
Epicentrarchus labrax
Pagellus bellottii
Pagellus erythrinus
Boops boops
Pachymetopon blochii
Pagrus pagrus
Argyrozona argyrozona
Dormatoschistus minutus
Dormatoschistus microps
Sobius niger
Solea solea
Pleuronectes platessa
Platichthys flesus
Limanda limanda
Microstomus pacificus
Hippoglossus stenolepis
Synnaptistes marmoratus
Sebastes mentella
Sebastes diutus
-2 -1 0 1 2
A


Variance decomposition
R2Max
SkR2k
Cumulative decomposition
DMax
SCE
B


phylogenetic correlogram
Moran's I
phylogenetic distance
C

5. Analysis of life history trait variation using orthonormal transform

auteurs

Sébastien ollier
Sandrine Pavoine
Daniel Chessel

résumé

In recent years, there has been an increased interest in studying the variability of a quantitative life history trait across a set of species sharing a common phylogeny. However, such studies have suffered from an insufficient development of statistical methods aiming at investigating into phylogenetic 'signal', i.e., autocorrelation of trait values with respect to the topological structure of the phylogenetic tree.

We present, here, a new approach that fits into the scheme of orthonormal transforms, like spectral and wavelet analyses for temporal data, in the sense that it expresses the topological properties of the phylogenetic tree via an orthonormal basis, which is further used to decompose the trait variance. Such a decomposition provides a structure function analogous to structure functions like periodogram, referred to as 'orthogram', which is relevant to characterise the phylogenetic signal in both graphical and statistical aspects. We also propose four complementary test statistics to be computed from orthogram values, that allow to diagnostic both intensity and nature of phylogenetic signal. The relevance of the method is illustrated by the analysis of many phylogenetic data sets, drawn from the literature and typifying contrasted levels and aspects of phylogenetic signal.

conferences

XXIInd International Biometric conference, 11-16 July 2004, Convention Center, Cairns Queensland Australia

Analysis of life history trait variation using orthonormal transform

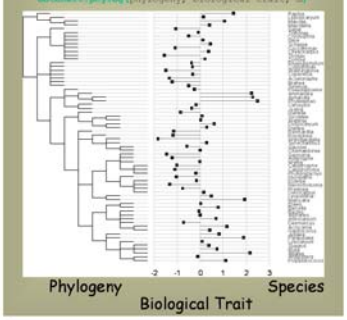
S. Ollier, S. Pavoine and D. Chessel
ollier@biomserv.univ-lyon1.fr
LBBE, UMR 5558, Claude Bernard University - Lyon1, 69 622 Villeurbanne Cedex France

XXIInd International Biometric conference, 11-16 July 2004, Cairns Convention Center, Cairns Queensland Australia

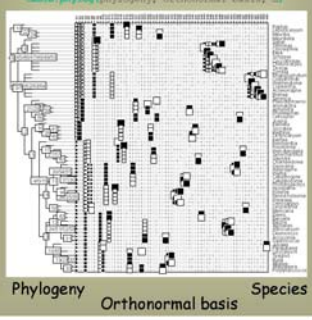
Introduction

In recent years, there has been an increased interest in studying the variation of a **quantitative life history trait** across a set of species sharing a common history[1]. The purpose of this study is to propose a relevant approach to describe and test how dependence between observations is distributed along a **phylogenetic tree**[2]. That method fits into the scheme of **orthonormal transforms**[3], like spectral and wavelet analyses for temporal data, in the sense that it expresses the topological properties of the phylogenetic tree via an **orthonormal basis**, which is further used to **decompose the trait variance**.

Graphical Display of Data



Canonical Orthonormal Basis



Orthonormal Transform Coefficients

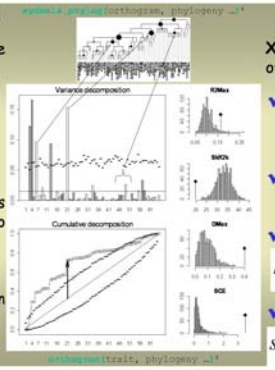
- Let **X** be the *N* column vector corresponding to the scaled biological trait
- Let **B** be the *N* by *N-1* matrix corresponding to the canonical orthonormal basis associated to the pylogeny
- We can analyse **X** with respect to **B** by computing the corresponding orthonormal transform coefficients: $\mathbf{R}=\mathbf{B}^T\mathbf{X}$
- It leads to the **decompostion of variance** and the **multilevel decomposition**

Decomposition of Variance

We can decompose the variance with respect to **B**:

$$\mathbf{X}^T\mathbf{X} = \mathbf{R}^T\mathbf{B}^T\mathbf{B}\mathbf{R} = \mathbf{R}^T\mathbf{R}$$

- Variance decomposition $\left(\mathbf{R}_i^2\right)_{1\le i\le N-1}$ against i
- Confidence enveloppes are built from Monte-Carlo randomizations of $\mathbf{X}_i$ values
- Cumulative decomposition $\left(\sum_{j=1}^{i}\mathbf{R}_j^2\right)_{1\le i\le N-1}$ against i

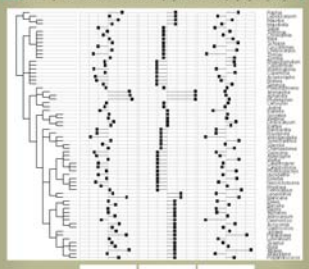


Non Parametric Tests

X is a simple random sample of the *N!* permutations (H0)

- $\mathrm{R2Max}(\mathbf{X}) = \max\left(R_1^2,\ldots,R_{N-1}^2\right)$
- $\mathrm{SkR2k}(\mathbf{X}) = \sum_{i=1}^{N-1} iR_i^2$
- $D\max(\mathbf{X}) = \max_{1\le m\le N-1}\left(\sum_{i=1}^{m}R_i^2 - \frac{m}{N-1}\right)$
- $SCE(\mathbf{X}) = \sum_i\left(\sum_{j=1}^{i}\mathbf{R}_j^2 - \sum_{j=1}^{i-1}\mathbf{R}_j^2\right)^2$

Multilevel Decomposition



References

[1] Harvey, P. H. and M. Pagel (1991). The Comparative Method in Evolutionary Biology, Oxford University Press.
[2] Ollier, S., *et al.* (submitted). Orthonormal transforms to detect and describe phylogenetic autocorrelation. Biometrics.
[3] Percival, D. B. and A. T. Walden (2000). Wavelet Methods for Time Series Analysis, Cambridge University Press.
[4] Ihaka, R. and R. Gentleman (1996). R: a language for data analysis and graphics. Journal of Computational and Graphical Statistics.

All computations and graphical displays involved in the preparation of this poster were carried out using R with routines of the **ade4** package available at http://pbil.univ-lyon1.fr.