Amanda Ferreira e Cunha

Doutoranda do curso de pós-graduação em Zoologia (IB) da Universidade de São Paulo.

O título do meu projeto é “Variabilidade e sinal filogenético da morfologia em organismos coloniais marinhos: um estudo com a família Campanulariidae (Cnidaria, Hydrozoa)”, orientada pelo Prof. Antonio Carlos Marques.

Meus exercícios

exec

Proposta de Trabalho Final

Plano A

Estou estudando a variabilidade morfológica em uma família de hidroides bentônicos (Cnidaria, Hydrozoa, Campanulariidae) cujas espécies e gêneros não são bem delimitados já que a maioria dos caracteres diagnósticos é muito variável. Para o estudo da variabilidade morfológica, é interessante compreender os padrões morfológicos considerando os diferentes níveis taxonômicos, desde o intraespecífico até o nível de gênero ou família. Nessa minha primeira proposta, pretendo implementar uma função no R que me permita uma análise da variabilidade dos caracteres morfológicos nesses diferentes níveis.

Dados de entrada: um data frame contendo as espécies nas linhas e os caracteres morfológicos (a princípio, somente dados contínuos de morfometria) nas colunas. A função deverá ser capaz de identificar os dados referentes ao nível intraespecífico (dados da mesma espécie), interespecífico (entre espécies de um mesmo gênero) e genérico (entre espécies de gêneros diferentes), para me gerar resultados específicos para cada um dos níveis.

Dados de saída: **1º** uma lista com os resultados da Análise de Componentes Principais para cada nível taxônomico (matriz de correlação, autovalores, autovetores, PCs); **2º** Biplots para cada nível taxonômico (talvez já com uma formatação gráfica padrão, para todos ficarem parecidos); **3º** também seria interessante se a função fizesse uma comparação entre as matrizes de correlação dos diferentes níveis taxonômicos para verificar diferenças nos padrões de correlação dos caracteres morfológicos em cada nível (com um teste de correlação de matrizes, por exemplo). Acho que para esse último resultado ficaria mais interessante se a função também tivesse como dado de entrada a filogenia ou uma matriz de distância filogenética para testar a estrutura filogenética de correlação das matrizes. Mas ainda não sei se isso tornaria a função muito complexa para ser executada.

Plano B

Outra análise interessante é relacionar a morfologia com a filogenia, verificando se determinados caracteres morfológicos têm sinal filogenético, ou seja, podem ser usados para a delimitação de linhagens. Como, em geral, muitos caracteres são usados para estudar a morfologia das espécies, seria interessante uma função que determinasse quais caracteres devem ser usados para testar o sinal filogenético. Nessa segunda proposta, pretendo implementar uma função que deverá gerar uma Análise de Componentes Principais com os dados de um data frame (como o anterior), e a partir das informações dos coeficientes de estrutura (loadings), selecionar as variáveis que tiveram maior importância para os primeiros eixos de ordenação, que descrevem a maior parte da variação dos dados (talvez aplicar uma método de seleção de eixos como Broken stick para determinar quantos eixos devem ser usados). Depois disso a função deverá realizar um teste de sinal filogenético (como o i de Moran ou PVR, por exemplo) somente com as variáveis que foram selecionadas anteriormente.

Dados de entrada: uma filogenia com informações de comprimento de ramos e um data frame com os mesmos táxons que compreendem os terminais da filogenia nas linhas e os caracteres morfológicos (a princípio, somente dados contínuos) nas colunas.

Dados de saída: um data frame contendo somente as variáveis selecionadas (espécies nas linhas e caracteres morfológicos nas colunas) e uma lista com os resultados das análises de sinal filogenético para cada variável selecionada.

Comentários

Ola Amanda,

Acho que a proposta A está mais simples e factível como proposta final do que a B. Então sugiro investir nela. Me parece que a variação da morfologia por nível taxonômico ou qualquer outro agrupamento é um problema relativamente geral na biologia e a tua função deve ser útil para um grupo maior de pessoas. Não esqueça de deixar a generalidade da função mais explícita.

Então um primeiro passo seria definir mais claramente o que está dentro das tuas análises de variabilidade (somente o PCA?). Depois basta definir como a função fará as análises em diferentes níveis dos teus dados. Você precisará de outras colunas no data-frame, identificando outros níveis taxonômicos como o gênero e a família, certo?

Resposta - Comentários

Olá, obrigada pelas sugestões. Sim, eu estava pensando em incluir nas análises de variabilidade o PCA e, em seguida, o teste de Mantel para comparar as matrizes de correlação das variáveis morfométricas de cada nível taxonômico. Definir a segunda parte (como a função fará as análises em diferentes níveis) está sendo um pouco difícil, mas acho que estou avançando. Eu acho que estou pensando nesse caminho que você sugeriu, acho que assim vai ser possível. Talvez incluir uma coluna no data-frame com números indicando cada nível taxonômico, ou criar listas com os dados de cada nível taxonômico em data-frames ou matrizes diferentes, talvez assim fique mais fácil de manipulá-los. Obrigada!

Código da Função (Plano A)

```
varimorph<-function (x, ngroup=1, inclusive=FALSE, scale.pca=TRUE,
scaling.biplot=2, choices.biplot=c(1,2), Mantel=FALSE, nsim=1000, x.cor,
y.cor, x.ncol, y.ncol)
{
  #carrega o pacote vegan.
  require("vegan")
  #Se o número de agrupamentos for menor ou igual a 1 não é possível
executar a função quando inclusive=TRUE, por isso aparece no console uma
mensagem de erro.
  if (ngroup<=1 & inclusive==TRUE)
  {
    return(cat("Atenção! Para inclusive=TRUE, ngroup deve ser maior que
1."))
  }
  #Se as colunas com os agrupamentos não são fatores, elas são transformadas
em fatores para que a função funcione corretamente.
  if (is.factor(unlist(x[,1:ngroup]))==FALSE)
  {
    for (i in 1:ngroup)
    {
      x[,i]<-as.factor(x[,i])
    }
  }
  #Se os agrupamentos não são inclusivos a função calcula uma PCA com todos
os dados, só muda as cores nos biplots de acordo com os fatores dos
agrupamentos.
  if(inclusive==FALSE)
  {
    #Calcula uma PCA com as colunas de dados contínuos (variáveis),
retirando as colunas de dados categóricos (os agrupamentos).
    pca<-rda(x[,(ngroup+1):dim(x)[2]], scale=scale.pca)
    resultado.pca<-summary(pca, scaling=scaling.biplot)
    #cria listas vazias para salvar os resultados do loop. O número de
elementos das listas é igual ao número de agrupamentos dados pelo usuário.
    grupos<-vector("list", ngroup)
    grupos2<-vector("list", ngroup)
    #Faz um loop para gerar listas com somente os nomes e as posiçoes dos
dados categóricos (agrupamentos), que serão usados para colorir os
    #biplots.
    for (i in 1:ngroup)
    {
      grupos[[i]]<-factor(x[,i])
      grupos2[[i]]<-as.numeric(factor(x[,i]))
    }
    #Cria um vetor com o nome das colunas de dados contínuos (variáveis).
    vetores<-colnames(x[(ngroup+1):dim(x)[2]])
    #Cria vetores com os resultados da pca referente aos escores ou
```

```
componentes principais (display="sites"), e aos autovetores
(display="species"). A função "scores"
    #é uma função para objetos da classe "rda". Para mais informações veja
?plot.cca .
    escores<-scores(resultado.pca, display="sites", choices=choices.biplot,
scaling=scaling.biplot)
    autovetores<-scores(resultado.pca, display="species",
choices=choices.biplot, scaling=scaling.biplot)
    #Abre um dispositivo tiff para salvar os biplots.
    tiff(file="myplot%03d.tif", width=15, height=15, unit="cm", res=100)
    #Muda as margens dos biplots que serão gerados (para entrar a legenda).
    par(mar=c(7,4,4,2))
    #Faz um loop para gerar os biplots de cada agrupamento.
    for (i in 1:ngroup)
    {
      #Cria um biplot vazio. Essa também é uma função para objetos da classe
"rda" (ver ?plot.cca para mais detalhes).
      plot(pca, display=c("sites","species"), type="n",
choices=choices.biplot, scaling=scaling.biplot,
main=paste("PCA",colnames(x)[i],"scaling",scaling.biplot))
      #Para cada biplot é feito outro loop para colorir os pontos de acordo
com os fatores de cada agrupamento.
      for (m in 1:nlevels(grupos[[i]]))
      {
        pontos<-points(escores[grupos2[[i]]==m,], pch=16, cex=1.5, col=m+1)
      }
      #Desenha as setas dos autovetores.
      arrows(0, 0, autovetores[,1], autovetores[,2], length=0.07, angle=20,
col="black")
      #Coloca o nome das variáveis originais, referente a cada autovetor.
      text(autovetores[,1], autovetores[,2], labels=vetores, cex=0.5, pos=4,
col="black")
      #Coloca uma legenda com as cores e os nomes dos fatores da cada
biplot.
      legend("bottom", paste(levels(grupos[[i]])), pch=16,
col=1+c(1:nlevels(grupos[[i]])), pt.cex=1.5, horiz=TRUE, xpd=TRUE,
inset=-0.33)
    }
    #Fecha o dispositivo tiff.
    dev.off()
    #Volta a configuração de margens de gráficos para o default.
    par(mar=c(5,4,4,2))
  }
  #Se os agrupamentos são inclusivos (níveis taxonômicos), é calculada uma
PCA para cada fator do agrupamento (a partir do segundo). Por isso é preciso
separar nos dados
  #originais os dados referentes a cada fator de cada agrupamento. As
porções dos dados extraídas dos dados originais serão a partir daqui
chamadas de 'partições'.
  if (inclusive==TRUE)
```

```
  {
    #cria listas vazias para salvar os resultados do loop. O número de
elementos das listas é igual ao número de fatores dos agrupamentos mais um,
que inclui a análise
    #com todos os dados.
    x.por.grupo<-vector("list",nlevels(unlist(x[,2:(ngroup)]))+1)
    grupos<-vector("list",nlevels(unlist(x[,2:ngroup]))+1)
    grupos2<-vector("list",nlevels(unlist(x[,2:ngroup]))+1)
    #Gera listas com os dados que serão utilizados para fazer as PCAs e os
biplots.A primeira e segunda listas geradas são os nomes e posição
    #(respectivamente) dos fatores de cada partição dos dados. A terceira
lista gerada é a lista das partições.
    for (i in 2:ngroup)
    {
      for (m in 1:nlevels(x[,i]))
      {

        #Como m pode ser 1 mais de uma vez (a cada vez que muda o i), para
não sobrescrever os dados de um loop anterior, os dados gerados são salvados
sempre na
        #primeira lista vazia de cada loop.
        grupos[grupos=="NULL"][[1]]<-
factor(x[x[,i]==levels(x[,i])[m],(i-1)])
        grupos2[grupos2=="NULL"][[1]]<-
as.numeric(factor(x[x[,i]==levels(x[,i])[m],(i-1)]))
        x.por.grupo[x.por.grupo=="NULL"][[1]]<-
x[x[,i]==levels(x[,i])[m],(ngroup+1):dim(x)[2]]
      }
    }
    #Acrescenta nas três listas (grupos=nomes dos fatores; grupos2=posição
dos fatores; x.por.grupo=partições) as informações referentes ao último
agrupamento (a última
    #coluna de agrupamentos) que inclui todos os dados originais.
    grupos[[length(grupos)]]<-factor(x[,ngroup])
    grupos2[[length(grupos2)]]<-as.numeric(factor(x[, ngroup]))
    x.por.grupo[[length(x.por.grupo)]]<-x[,(ngroup+1):dim(x)[2]]

    #Cria listas vazias para salvar os resultados do loop. O número de
elementos das listas é igual ao número de fatores dos agrupamentos (a partir
da segunda coluna)
    #mais um, que inclui a análise com todos os dados.
    pca<-vector("list",nlevels(unlist(x[,2:ngroup]))+1)
    resultado.pca<-vector("list",nlevels(unlist(x[,2:ngroup]))+1)
    #Gera listas com os resultados da pca para cada partição dos dados.
    for (i in 1:length(x.por.grupo))
    {
      pca[[i]]<-rda(x.por.grupo[[i]], scale=scale.pca)
      resultado.pca[[i]]<-summary(pca[[i]], scaling=scaling.biplot)
    }

    #Nomeia cada uma das listas com os mesmos nomes dos fatores dos
```

```
agrupamentos, menos a última lista, que é nomeada depois com o mesmo nome da
última coluna de
    #agrupamentos.
    names(resultado.pca)<-levels(unlist(x[,2:ngroup]))
    names(resultado.pca)[length(resultado.pca)]<-names(x[ngroup])
    #Cria um vetor com o nome das colunas de dados contínuos (variáveis).
    vetores<-colnames(x[,(ngroup+1):dim(x)[2]])

    #Abre um dispositivo tiff para salvar os biplots.
    tiff(file="myplot%03d.tif", width=15, height=15, unit="cm", res=100)
    #Muda as margens dos biplots que serão gerados (para entrar a legenda).
    par(mar=c(7,4,4,2))

    ##Faz um loop para gerar os biplots de cada partição dos dados.
    for (i in 1:length(x.por.grupo))
    {
      #Cria vetores com os resultados da pca referente aos escores ou
componentes principais (display="sites"), e aos autovetores
(display="species") de cada partição
      #dos dados. A função "scores" é uma função para objetos da classe
"rda". Para mais informações veja ?plot.cca .
      escores<-scores(resultado.pca[[i]], display="sites",
choices=choices.biplot, scaling=scaling.biplot)
      autovetores<-scores(resultado.pca[[i]], display="species",
choices=choices.biplot, scaling=scaling.biplot)
      #Cria um biplot vazio para cada partição dos dados. Essa também é uma
função para objetos da classe "rda" (ver ?plot.cca para mais detalhes).
      plot(pca[[i]], display=c("sites","species"), type="n",
choices=choices.biplot, scaling=scaling.biplot,
main=paste("PCA",names(resultado.pca[i]),"scaling",scaling.biplot))
      #Para cada biplot é feito outro loop para colorir os pontos de acordo
com os fatores de cada partição dos dados.
      for (m in 1:nlevels(grupos[[i]]))
      {
        pontos<-points(escores[grupos2[[i]]==m,], pch=16, cex=1.5, col=m+1)
      }
      #Desenha as setas dos autovetores.
      arrows(0, 0, autovetores[,1], autovetores[,2], length=0.07, angle=20,
col="black")

      #Coloca o nome das variáveis originais, referente a cada autovetor.
      text(autovetores, labels=vetores, cex=0.5, pos=4, col="black")
      #Coloca uma legenda com as cores e os nomes dos fatores de cada
biplot.
      legend("bottom", paste(levels(grupos[[i]])), pch=16,
col=1+c(1:nlevels(grupos[[i]])), pt.cex=1.5, horiz=TRUE, xpd=TRUE,
inset=-0.33)
    }
    #Fecha o dispositivo tiff.
    dev.off()
```

```
    #Volta a configuração de margens de gráficos para o default.
    par(mar=c(5,4,4,2))
  }

  if (Mantel==TRUE)
  {
    #Seleciona os dados referentes ao fatores informados pelo usuário, a
partir dos dados originais.
    x.dados<-x[x[x.ncol]==x.cor,(ngroup+1):dim(x)[2]]
    y.dados<-x[x[y.ncol]==y.cor,(ngroup+1):dim(x)[2]]
    #Gera uma matriz de correlação com os dados de cada fator.
    x.corr<-cor(x.dados)
    y.corr<-cor(y.dados)
    #Seleciona somente os dados abaixo da diagonal da matriz de correlação
de cada fator, e transforma esses dados em um vetor.
    x.cor.vetor<-as.vector(as.dist(x.corr))
    y.cor.vetor<-as.vector(as.dist(y.corr))
    #Calcula os desvios de cada elemento de cada vetor.
    desv.x<-x.cor.vetor-mean(x.cor.vetor)
    desv.y<-y.cor.vetor-mean(y.cor.vetor)
    #Calcula o produto entre os elementos correspondentes de cada vetor, e
soma os produtos.
    cross.prod<-sum(desv.x*desv.y)
    #Calcula a raiz quadrada do produto das somas dos desvios quadráticos de
cada vetor.
    denom<-sqrt(sum((x.cor.vetor-mean(x.cor.vetor))^2)*(sum((y.cor.vetor-
mean(y.cor.vetor))^2)))
    #Calcula a estatística r de Mantel.
    rm<-cross.prod/denom
    #Cria um vetor vazio para salvar os resultados das simulações, com o
número de elementos igual ao número de simulações definidas pelo usuário.
    rm.aleat<-rep(NA, nsim)

    #Coloca o valor observado da estatística r de Mantel na primeira posição
do vetor de simulações.
    rm.aleat[1]<-rm
    #Faz um loop para realizar o número de simulações definidas pelo
usuário, criando uma distribuição nula da estatística r de Mantel.
    for(i in 2:nsim)
    {
      #Gera um novo vetor de dados x (o y permanece o mesmo), aleatorizando
os elementos correspondentes às colunas dos dados originais.
      x.dados2<-apply(x.dados,2,sample)
      x.corr2<-cor(x.dados2)
      x.cor.vetor2<-as.vector(as.dist(x.corr2))
      #Faz novamente os cálculos da estatística r de Mantel, agora com o
novo vetor de dados aleatorizados.
      desv.x<-x.cor.vetor2-mean(x.cor.vetor2)
      desv.y<-y.cor.vetor-mean(y.cor.vetor)
      cross.prod<-sum(desv.x*desv.y)
      denom<-sqrt(sum((x.cor.vetor2-
```

```
mean(x.cor.vetor2))^2)*(sum((y.cor.vetor-mean(y.cor.vetor))^2)))
      rm.aleat[i]<-cross.prod/denom
    }
    #Faz um teste bicaudal para testar a significância do resultado
observado. Hipótese alternativa: a correlação entre as duas matrizes é
diferente de zero
    #(independe do sinal da correlação). Para isso, calcula o número de
vezes que o módulo da estatística r simulada foi maior ou igual ao módulo da
estatística
    #r observada, e divide pelo número de simulações.
    rm.maior.ou.menor<-sum(abs(rm.aleat)>=abs(rm))
    p<-rm.maior.ou.menor/nsim
    #Retorna no console os resultados do teste de Mantel.
    cat("\n\nResultado do Teste de Mantel\n\t","\n\n\t Estatística r de
Mantel:", round(rm,4), "\n\n\t Valor de p baseado em",nsim,"permutações:",
p, "\n", "\n\nResultado da PCA\n\n\t")
  }
    #Retorna no console os resultados da pca.
    return(resultado.pca)
}
```

Página de Ajuda da Função

```
varimorph                package:unknown                R Documentation


Análises de Componentes Principais (PCA) em diferentes níveis taxonômicos
(ou outro agrupamento) e Teste de Mantel.


Description:

    A função realiza Análises de Componentes Principais considerando
diferentes níveis de análise ou agrupamentos definidos previamente pelo
usuário. A partir
de um data frame contendo todos os dados, a função identifica os diferentes
níveis de análise determinados pelo usuário,  realizando a PCA e construindo
gráficos do
tipo biplot para cada nível de análise. Além disso, a função também oferece
a opção de realizar o Teste de Mantel para comparar as matrizes de
correlação de diferentes
subconjuntos dos dados originais, definidos pelo usuário.

Usage:

    varimorph (x, ngroup=1, inclusive=FALSE, scale.pca=TRUE,
scaling.biplot=2, choices.biplot=c(1,2), Mantel=FALSE, nsim=1000, x.cor,
```

```
y.cor, x.ncol, y.ncol)

Arguments:

        um data frame contendo os dados para as análises. As variáveis devem
estar nas colunas e as observações nas linhas. Os níveis de análise devem
estar
    x    discriminados nas primeiras colunas da matriz de dados. Cada
agrupamento é uma coluna e os fatores dessa coluna determinam os níveis do
agrupamento. A
                primeira linha do data frame deve conter o nome das colunas.
Veja 'Details' para mais detalhes.

      ngroup    número de colunas que identificam os agrupamentos no data
frame. Quando inclusive=TRUE, ngroup deve ser maior que 1.
    inclusive   se igual a TRUE, determina que os agrupamentos definidos
pelo usuário são inclusivos, ou seja, representam níveis hierárquicos. Se
inclusive=FALSE
        (default), os agrupamentos não representam níveis hierárquicos. Veja
'details' para mais detalhes.

    scale.pca   se igual a TRUE, padroniza as variáveis a partir da matriz
de correlação dos dados. É importante quando as variáveis não se encontram
na mesma unidade
        de escala. Veja 'details' para mais detalhes.
        se refere a forma como os objetos e variáveis serão mostrados juntos
no biplot da PCA. Se igual a 1, biplot de distância, as distâncias entre os
objetos
 scaling.biplot são aproximações das suas distâncias euclidianas no espaço
multidimensional e os ângulos entre os vetores não têm sentido analítico; Se
igual a 2,
        biplot de correlação, as distâncias entre os objetos não são
aproximações das suas distâncias euclidianas no espaço multidimensional e os
ângulos entre
        os vetores refletem suas correlações. Para mais detalhes, veja o
argumento "scaling" em ?summary.cca (pacote vegan).

choices. biplot define os eixos do biplot, ou seja, quais componentes
principais (PCs) devem ser usados para construir o biplot. O default,
c(1,2), significa que o
        biplot será construído com o primeiro PC no eixo x e o segundo PC no
eixo y.

    Mantel  se igual a TRUE, além da PCA é realizado um Teste de Mantel
entre as matrizes de correlação dos dados definidos pelo usuário em x.cor e
y.cor.

    nsim    número de simulações que será utilizado na construção do modelo
nulo para o teste de significância da Estatística r de Mantel.

  x.cor, y.cor  fatores dos agrupamentos definidos pelo usuário cujas
```

```
matrizes de correlação serão comparadas pelo Teste de Mantel. Devem se
referir a subconjuntos dos
        dados originais. Veja 'details' para mais detalhes.

 x.ncol, y.ncol números das colunas dos fatores definidos em x.cor e y.cor.


Details:

        A Análise de Componentes Principais é realizada utilizando-se a
função rda do pacote vegan, por isso para executar a função varimorph é
preciso ter esse pacote
instalado. Os dados de entrada da função devem ser do tipo data frame, e a
primeira linha deve conter os nomes das colunas. Esses nomes serão
utilizados para
identificar os resultados das análises e os biplots de cada agrupamento. Os
agrupamentos definidos pelo usuário devem estar identificados nas primeiras
colunas do data
frame. Cada coluna deve ser um agrupamento, e os fatores da coluna devem se
referir aos níveis dos agrupamentos. Quando os agrupamentos são níveis
taxonômicos (ou
qualquer outro tipo de agrupamento que represente níveis hierárquicos), as
primeiras colunas do data frame que identificam os agrupamentos devem
conter, nesse ordem,
os agrupamentos menos inclusivos até os mais inclusivos (de espécie até
família, ou de subespécie até gênero, etc). Nesse caso, o argumento
inclusive deve ser igual a
TRUE. Quando os agrupamentos não representam níveis hierárquicos (por
exemplo, agrupamento por local de ocorrência e agrupamento por período de
ocorrência), eles podem
estar dispostos em qualquer ordem, desde que nas primeiras colunas do data
frame. Nesse caso, o argumento inclusive deve ser igual a FALSE. Veja o
arquivo "Exemplo.csv"
para um modelo do data frame.
    Quando inclusive=TRUE, a Análise de Componentes Principais é realizada
considerando-se cada fator do agrupamento, a partir da segunda coluna de
agrupamentos.
Assim, se os agrupamentos foram definidos como espécies, gêneros e famílias,
as análises são feitas considerando os dados referentes às espécies do
gênero 1, às
espécies do gênero 2, e assim por diante; os gêneros da família 1, os
gêneros da família 2, e assim por diante; e por final, as diferentes
famílias. Quando inclusive=
FALSE, uma Análise de Componentes Principais é realizada com todos os dados,
e um biplot é construído para cada agrupamento. Assim, para que a função
funcione
corretamente, é preciso que o número de agrupamentos informado (ngroup) seja
exatamente igual ao do data frame. Para realizar a função com menos
agrupamentos que
aqueles definidos no data frame, o usuário deve elaborar uma nova planilha
```

de dados, retirando os agrupamentos indesejados, e criar um novo data frame para esses dados.

Os argumentos referentes à Análise de Componentes Principais são os mesmos da função rda do pacote vegan, e mais detalhes sobre esses argumentos podem ser
consultados em ?rda e ?summary.cca (como 'scale', 'scaling' e 'choices'). Jolliffe (2002, pág. 42) discute sobre o uso de matrizes de correlação ou covariância para a
obtenção dos componentes principais. Em geral, quando a matriz de covariância é usada com variáveis que se encontram em escalas diferentes (principalmente quando suas
variâncias são muitos diferentes), os primeiros componentes representam basicamente as diferenças relativas nas variâncias, contendo pouca informação sobre a variação
dos dados. Nesses casos, a matriz de correlação é mais indicada (scale.pca=TRUE). O arquivo "Exemplo.csv" contém um exemplo de dados em que duas variáveis (V1 e V12)
apresentam variâncias muito maiores que as demais variáveis. Veja o que acontece com seus respectivos autovalores e autovetores quando scale.pca=FALSE (em 'Examples').
O tamanho dos vetores do biplot também é influenciado por essas duas variáveis. Em relação ao argumento scaling.pca, veja Legendre & Legendre (1998, pág. 403) e Borcard et al. (2011,
pág.120) para mais detalhes sobre as diferenças na forma como os biplots são contruídos quando scaling.pca=1 ou 2.

Na função varimorph, o Teste de Mantel é utilizado para verificar se os padrões de correlação das variáveis muda entre os diferentes fatores dos agrupamentos. Para isso, o usuário deve definir pelos argumentos x.cor e y.cor quais fatores se deseja comparar. Para que o Teste de Mantel seja válido, as variáveis
utilizadas para construir a matriz X e a matriz Y devem ser geradas a partir de diferentes conjuntos de dados (veja Legendre & Legendre 1998 para mais detalhes). Por
esse motivo, x.cor e y.cor devem se referir a fatores dos agrupamentos, que determinam diferentes subconjuntos dos dados originais. Assim, para cada subconjunto dos
dados é construída uma matriz de correlação, e essas matrizes são comparadas por meio do Teste de Mantel. O teste de significância é construído a partir de um modelo
nulo, por meio de permutações dos dados originais.

Value:

A função varimorph retorna os seguintes resultados:

Resultados da PCA objeto da classe rda, contendo os autovalores, autovetores (species scores), e componentes principais (site scores). Para mais detalhes
veja ?cca.object, do pacote vegan.

```
     Biplots         gráficos do tipo biplot no formato tiff (15cm de
altura, 15cm de diâmetro e resolução de 100 ppi). Os gráficos gerados pelas
análises
                são salvos diretamente no diretório de trabalho do usuário.

 Resultados do Teste de Mantel  quando Mantel=TRUE, além dos resultados da
PCA, a função retorna o valor da Estatística r de Mantel e o valor de p,
obtido pelo teste
                de significância.

Author(s):

        Amanda Ferreira e Cunha.

References:

        Legendre, P. & Legendre, L. 1998. Numerical Ecology. Second English
Edition, Elsevier Science B.V., Amsterdam, 853p.
    Jolliffe, I.T. 2002. Principal Components Analysis. Second Edition,
Springer Series in Statistics, 457p.
    Borcard, D.; Gillet, F. & Legendre, P. 2011. Numerical Ecology with R.
Springer, 306p.


See Also:

     ?rda, ?summary.cca, ?object.cca e ?plot.cca, do pacote vegan.

Examples:

        #Para ler o arquivo de dados "Exemplo.csv"
    dados<-read.table("Exemplo.csv", header=TRUE, sep=";")

    #Carrega a função varimorph
    source("varimorph.R")

    #Executa a PCA considerando os agrupamentos como níveis hierárquicos
    varimorph(dados, ngroup=3, inclusive=TRUE)

    #Executa a PCA (com outra forma de construção dos biplots) e o Teste de
Mantel com os dados dos fatores "C.agas" e "C. tincta", da coluna 2
(Espécies)
    varimorph(dados, ngroup=3, inclusive=TRUE, scaling.biplot=1,
Mantel=TRUE, x.cor="C.agas", y.cor="C.tincta", x.ncol=2, y.ncol=2)

    #Executa a PCA considerando os agrupamentos como não inclusivos e plota
os biplots com o segundo e terceiro componentes principais
    varimorph(dados, ngroup=3, choices.biplot=c(2,3))

    #Executa a PCA considerando os agrupamentos como não inclusivos, plota
```

```
os biplots com o primeiro e segundo componentes principais (default), e
executa o Teste
    #de Mantel com os dados dos fatores "Campanularia" e "Orthopyxis", da
coluna 3 (Gêneros)
    varimorph(dados, ngroup=3, Mantel=TRUE, x.cor="Campanularia",
y.cor="Orthopyxis", x.ncol=3, y.ncol=3)

    #Não executa, ngroup deve ser igual ao número de agrupamentos dos dados
(3, nesse caso, populações, espécies e gêneros)
    varimorph(dados, ngroup=1) #Para executar a função com menos
agrupamentos, o data frame deve ser alterado

    #Executa a PCA com os dados não padronizados (não recomendado para esses
dados, já que as variáveis apresentam variâncias muito discrepantes)
    varimorph(dados, ngroup=3, scale.pca=FALSE)
```

Arquivos da Função

varimorph.r pagina_de_ajuda_varimorph.txt exemplo.csv