Package 'EcoGenetics'

July 11, 2015

Type Package

Title Spatial Analysis of Phenotypic, Genotypic and Environmental Data

Version 1.2.0-2

Date 2015-07-11

Author Leandro Roser, Juan Vilardi, Beatriz Saidman and Laura Ferreyra

Maintainer Leandro Roser <learoser@gmail.com>

Description Geostatistical tools for analyzing spatial patterns in population biology. Easy integration of information from multiple sources with ``ecogen'' objects.

License GPL (>= 2)

URL https://github.com/cran/EcoGenetics

LazyLoad yes

Depends R (>= 3.0), methods

Imports ggplot2, party, raster, reshape2, rgdal, rkt, SoDA, sp,

Suggests adegenet

Collate 'ZZZ.R' 'auxiliar.R' 'int.genind.R' 'ecogen.1OF5.definition.R' 'ecogen.2OF5.constructor.R' 'ecogen.3OF5.basic.methods.R' 'ecogen.4OF5.brackets.R' 'ecogen.5OF5.get&set.R' 'accesors.R' 'classes.R' 'control.R' 'deprecated.R' 'eco.2geneland.R' 'eco.2genepop.R' 'eco.2gstudio.R' 'eco.2hierfstat.R' 'eco.2spagedi.R' 'eco.NDVI.R' 'eco.NDVI.post.R' 'eco.alfreq.R' 'eco.association.R' 'eco.cbind.R' 'eco.clear.R' 'eco.convert.R' 'eco.cormantel.R' 'eco.correlog.R' 'eco.detrend.R' 'eco.forestplot.R' 'eco.format.R' 'eco.genepop2df.R' 'eco.gsa.R' 'eco.kin.loiselle.R' 'eco.lagweight.R' 'eco.lmtree.R' 'eco.lsa.R' 'eco.malecot.R' 'eco.mantel.R' 'eco.merge.R' 'eco.order.R' 'eco.pairtest.R' 'eco.post.geneland.R' 'eco.rankplot.R' 'eco.rbind.R' 'eco.remove.R' 'eco.subset.R' 'eco.theilsen.R' 'eco.variogram.R' 'eco.weight.R' 'int.break.R' 'int.convert.R' 'int.crosscor.R' 'int.geary.R' 'int.jackknife.R' 'int.joincount.R' 'int.kin.loiselle.R' 'int.mantel.R'

'int.moran.R' 'int.multitable.R' 'int.random.test.R'
'miscellaneous.R' 'plot.methods.R' 'show_summary.methods.R'

NeedsCompilation no

Repository CRAN

Date/Publication 2015-07-11 18:31:53

R topics documented:

`EcoGenetics-package` *Spatial Analysis of Phenotypic, Genotypic and Environmental Data*

Description

EcoGenetics provides geostatistical tools for the spatial analysis of phenotypic, genotypic and environmental data. The package has been designed to make easy the storing, handling and integration of the available information from multiple sources, under an S4 philosophy.

Details

Package:	EcoGenetics
Type:	Package
Version:	1.2.0-2
Date:	2015-07-11
License:	GPL (>=2)

———————— o ————————

I. STRUCTURE OF THE PACKAGE

EcoGenetics has four basic modules. The **base module** is composed by general functions (multiple lm, detrending spatial data utility, etc.). The **general spatial module** computes global (Moran's I, Mantel test, etc.) and local (Getis-Ord's G*, local Moran's I, etc.) spatial tests. These analyses use a spatial weights matrix, provided by the **spatial weights module**. The **lag analysis module** performs two basic analyses: the obtention of variograms and correlograms (see also this link). This module uses the other tool provided by the spatial weights module: a sequence of spatial weights matrices.

The package have also special plot methods, as rankplots and forestplots (implemented here for local spatial analysis), conversors of data to other programs, as (genepop - an importer tool is also defined for genepop-, SPAGeDi, etc.). Basic manipulation of genetic matrices is allowed by

eco.convert and eco.format. Tools for computation of NDVI in Landsat imaginery, post-process of rasters and temporal analysis can be found in eco.NDVI, eco.NDVI.post and eco.theilsen. Other useful functions are aue.sort (for ordering alleles), eco.alfreq (it computes histograms of allelic frequencies for detection of bottlenecks), eco.post.geneland and eco.pairtest.

The results obtained with the main functions defined in EcoGenetics are object of class S4. As default characteristic of the package design, these objects have a "show"" method for a general overview of the results, and methods to extract the results stored in slots (generic accessors and double square brackets ("[[") definitions).

For storing and pre-processing the data to analyze, the package defines a special class: the class ecogen.

II. STRUCTURE OF ECOGEN OBJECTS: HANDLING AND INTEGRATING INFORMATION

Ecological genetics research requires the integration of data originated in different sources. The class ecogen has been designed for handling multidimensional data. Its basic structure is the following:

- A **XY** slot, storing a data frame with geographic coordinates.

- A **P** slot, storing a phenotypic data frame.

- A **G** slot, storing a genotypic data frame.

- An **A** slot,containing as frequencies (of alleles in codominant data) the information of G.

- An **E** slot, storing an environmental data frame.

- A **S** slot, storing a data frame with classes assigned to the individuals.

- A **C** slot, for a custom data frame.

- An **OUT** slot, containing a list for the storage of the results.

III. A BRIEF OVERVIEW OF THE DATA HANDLING METHODS DEFINED FOR ECOGEN OBJECTS

The construction of a new "ecogen" object from a data frame is made with the homonymous function.

```
library("EcoGenetics")

data(eco.test)

eco <- ecogen(XY = coordinates, P = phenotype, G = genotype, E = environment, S = structure,
order.G = TRUE)

# The following methods can be used with ecogen objects:

# - - - - - -

# 1. subsetting by row-method, using single square brackets ("[")

eco.sub <- eco[1:50]

# 2. merging-method, for two objects

eco1 <- eco

merged <- eco.merge(eco, eco1)

# 3. subsetting-method, in reference to a group in the S slot (in this case, "1")

eco.subS <- eco.subset(eco,"pop", 1)
```

 # 4. binding by row-method (duplicated row names not allowed)

eco2 <- eco; rownames(eco2[["P"]]) <-226:450

eco.r <- eco.rbind(eco, eco2)

 # 5. binding by column-method

eco.c <- eco.cbind(eco, eco1)

 # 6. ordering by row-method, using the rows in the XY data frame as reference

ecoslot.P(eco1) <- eco[["P"]][sample(1:173),] ## object with unordered rows in P data frame

ordered <- eco.order(eco1)

 # 7. get-method using generic accessors and "[[" (equivalent methods):

ecoslot.XY(eco) ; eco[["XY"]]

ecoslot.P(eco); eco[["P"]]

ecoslot.A(eco); eco[["A"]]

ecoslot.E(eco); eco[["E"]]

ecoslot.S(eco); eco[["S"]]

ecoslot.C(eco); eco[["S"]]

ecoslot.OUT(eco); eco[["OUT"]]

 # 8. set-method using accessors and "[[" (equivalent methods):

eco.temp <- ecogen(XY = coordinates, P = phenotype)

eco.temp

ecoslot.G(eco.temp, order.G = TRUE) <- genotype

this is equivalent, in square brackets notation, to:

eco[["G", order.G=TRUE]] <- genotype

ecoslot.E(eco.temp) <- environment

identical to eco[["E"]] <- environment

ecoslot.S(eco.temp) <- structure

identical to eco[["S"]] <- structure

eco.temp

 # 9. appending-method (storing information generated via accessors)

fitting a multiple linear regression model:

linear.analysis <- eco.lmtree(eco[["P"]], eco[["E"]], "mlm")

storing the results:

ecoslot.OUT(eco) <- linear.analysis

eco

Storing multiple result at once:

a <- c(1:10)

b <- c(2:30)

ecoslot.OUT(eco) <- list(eco, a, b)

eco

the use of the accessor OUT has its equivalence in double square brackets notation:

eco[["OUT"]] <- list(eco, a, b)

summary table

ecoslot.OUT(eco)

the data frame shows the results stored in alphabetical order and their classes. The specification of a second name, return the corresponding stored object:

ecoslot.OUT(eco, "a")

note that the append method is a particular case of the ## set method with accessors / "[[" for the slot OUT.

 # 10. removing-method

removing objects a and b from eco

eco <- eco.remove(eco, a, b)

- - - - - -

Finally, the workspace can be cleared, only storing the desired object:

ls()

eco.clear(eco)

ls()

———————— o ————————

Author(s)

Leandro Roser, Juan Vilardi, Beatriz Saidman and Laura Ferreyra

Maintainer: Leandro Roser <leandroroser@ege.fcen.uba.ar>

References

Anselin L. 1995. Local indicators of spatial association-LISA. Geographical analysis. 27: 93-115.

Borcard D., F. Gillet, and P. Legendre. 2011. Numerical ecology with R. Springer Science & Business Media.

Chander G., B. Markham, and D. Helder. 2009. Summary of current radiometric calibration coefficients for Landsat MSS, TM, ETM+, and EO-1 ALI sensors. Remote sensing of environment, 113: 893-903.

Chavez P. 1989. Radiometric calibration of Landsat Thematic Mapper multispectral images. Photogrammetric Engineering and Remote Sensing, 55: 1285-1294.

Chavez P. 1996. Image-based atmospheric corrections-revisited and improved. Photogrammetric engineering and remote sensing, 62: 1025-1035.

Double M., R. Peakall, N. Beck, and Y. Cockburn. 2005. Dispersal, philopatry, and infidelity: dissecting local genetic structure in superb fairy-wrens (Malurs cyaneus). Evolution 59: 625-635.

Dray S., and A., Dufour. 2007. The ade4 package: implementing the duality diagram for ecologists. Journal of statistical software, 22: 1-20.

Freedman D., and P. Diaconis. 1981. On the histogram as a density estimator: L 2 theory. Probability theory and related fields, 57: 453-476.

Geary R. 1954. The contiguity ratio and statistical mapping. The incorporated statistician, 115-146.

Getis A., and J. Ord. 1992. The analysis of spatial association by use of distance statistics. Geographical analysis, 24: 189-206.

Goslee S. 2011. Analyzing remote sensing data in R: the landsat package. Journal of Statistical Software, 43: 1-25.

Goudet J. 2005. Hierfstat, a package for R to compute and test hierarchical F-statistics. Molecular Ecology Notes, 5: 184-186.

Guillot G., F. Mortier and A. Estoup. 2005. GENELAND: a computer package for landscape genetics. Molecular Ecology Notes, 5: 712-715.

Jombart T. 2008. adegenet: a R package for the multivariate analysis of genetic markers. Bioinformatics, 24: 1403-1405.

Kalisz S., J. Nason, F.M. Handazawa, and S. Tonsor. 2001. Spatial population genetic structure in Trillium grandiflorum: the roles of dispersal, mating, history, and selection. Evolution 55: 1560-1568.

Legendre P., and L. Legendre. 2012. Numerical ecology. Third English edition .Elsevier Science, Amsterdam, Netherlands.

Lichstein J., T. Simons, S. Shriner, and K. Franzreb. 2002. Spatial autocorrelation and autoregressive models in ecology. Ecological monographs, 72: 445-463.

Loiselle B., V. Sork, J. Nason, and C. Graham. 1995. Spatial genetic structure of a tropical understory shrub, Psychotria officinalis (Rubiaceae). American Journal of Botany 1420-1425.

Moran P. 1950. Notes on continuous stochastic phenomena. Biometrika, 17-23.

Oden N., and R. Sokal. 1986. Directional autocorrelation: an extension of spatial correlograms to two dimensions. Systematic Zoology, 35: 608-617.

Ord J., and A. Getis. 1995. Local spatial autocorrelation statistics: distributional issues and an application. Geographical analysis, 27: 286-306.

Reich R., R. Czaplewski and W. Bechtold. 1994. Spatial cross-correlation of undisturbed, natural shortleaf pine stands in northern Georgia. Environmental and Ecological Statistics, 1: 201-217.

Sokal R., and N. Oden 1978. Spatial autocorrelation in biology: 1. Methodology. Biological journal of the Linnean Society, 10: 199-228.

Sokal R., and N. Oden. 1978. Spatial autocorrelation in biology. 2. Some biological implications and four applications of evolutionary and ecological interest. Biological Journal of the Linnean Society, 10: 229-49.

Sokal R. 1979. Ecological parameters inferred from spatial correlograms. In: G. Patil and M. Rosenzweig, editors. Contemporary Quantitative Ecology and elated Ecometrics. International Co-operative Publishing House: Fairland, MD, pp. 167-96.

Sokal R. 1986. Spatial data analysis and historical processes. In: E. Diday, Y. Escoufier, L. Lebart, J. Pages, Y. Schektman, and R. Tomassone, editors. Data analysis and informatics, IV. North-Holland, Amsterdam, The Netherlands, pp. 29-43.

Sokal R., N. Oden and B. Thomson. 1998. Local spatial autocorrelation in a biological model. Geographical Analysis, 30: 331-354.

Sokal R., and B. Thomson. 2006. Population structure inferred by local spatial autocorrelation: an example from an Amerindian tribal population. American journal of physical anthropology, 129: 121-131.

Song C., C. Woodcock, K. Seto, M. Lenney and S. Macomber. 2001. Classification and change detection using Landsat TM data: when and how to correct atmospheric effects?. Remote sensing of Environment, 75: 230-244.

Sturges H. 1926. The choice of a class interval. Journal of the American Statistical Association, 21: 65-66.

Tucker C. 1979. Red and photographic infrared linear combinations for monitoring vegetation. Remote sensing of Environment, 8: 127-150.

Vekemans, X., and O. Hardy. 2004. New insights from fine-scale spatial genetic structure analyses in plant populations. Molecular Ecology, 13: 921-935.

Wu. 1986. Jackknife, bootstrap and other resampling methods in regression analysis. the Annals of Statistics, 1261-1295.

Examples

```
 ## Not run:


#---Detrending spatial data with polynomial interpolation---#


data(eco2)

## original data
data1 <- matrix(eco2[["P"]][,1], 30, 30)
image(data1)

## original data + trend
data2 <- matrix(eco2[["P"]][,2], 30, 30)
image(data2)

## data detrending
data2.det <- eco.detrend(Z = eco2[["P"]][,2], XY =  eco2[["XY"]], degree =  1)
data2.det <- ecoslot.RES(data2.det)
data2.det <- matrix(data2.det$df1, 30, 30)
image(data2.det)


#---Multiple Linear Regression fit---#


data(eco.test)
mymod <- "E1+E2*E3"
mod <- eco.lmtree(df1 = eco[["XY"]], df2 = eco[["E"]],
analysis = "mlm", mod.class = mymod)
summary(mod)
```

```
#---Multiple Conditional Inference Trees---#


data(eco.test)
mymod <- "E1+E2*E3"
mod <- eco.lmtree(df1 = eco[["P"]], df2 = eco[["E"]],
analysis = "mctree", mod.class = mymod, fact = eco[["S"]]$structure)
summary(mod)


#---Global spatial analysis---#


## Moran's I

### one test
data(eco.test)
con <- eco.weight(eco[["XY"]], method = "circle", d1 = 0, d2 = 2)
global <- eco.gsa(Z = eco[["P"]][, 1], con = con, , method = "I", nsim = 200)
global

require(adegenet)
con2<-chooseCN(eco[["XY"]], type = 1, result.type = "listw", plot.nb = FALSE)
global <- eco.gsa(Z = eco[["P"]][, 1], con = con2, , method = "I", nsim = 200)
global

### multiple tests
con <- eco.weight(eco[["XY"]], method = "circle", d1 = 0, d2 = 2)
global <- eco.gsa(Z = eco[["P"]], con = con, , method = "I", nsim = 200)
global


## Geary's C

data(eco.test)
global.C <- eco.gsa(Z = eco[["P"]][, 1], con = con, method = "C", nsim = 200)
global.C


## Bivariate's Moran's Ixy

data(eco.test)
global.Ixy <- eco.gsa(Z = eco[["P"]][, 1], Y = eco[["E"]][, 1],
con = con, method = "CC", nsim = 200)
global.Ixy


## Join-Count

data(eco.test)
global.JC <- eco.gsa(Z = 2* eco[["A"]][, 1], ncod = 1,
con = con, method = "JC", nsim = 5)
```

```
global.JC


# Mantel test

data(eco.test)
eco.mantel(d1 = dist(eco[["P"]]), d2 = dist(eco[["E"]]), nsim = 99)


## Partial Mantel test

data(eco.test)
eco.mantel(d1 = dist(eco[["P"]]), d2 = dist(eco[["E"]]),
dc = dist(eco[["XY"]]), nsim = 99)


#---Local spatial analysis---#


## Getis-Ord's G*

data(eco.test)
require(ggplot2)
con<- eco.weight(eco[["XY"]], method = "knearest",  k = 4, self = TRUE)
### self = TRUE for G*
getis.ak <- eco.lsa(eco[["P"]][, 1], con, method = "G*", nsim = 99,
adjust = "none")
getis.ak

### to plot the results, the function "eco.lsa" calls "eco.rankplot"
### (see ?eco.rankplot) when test = "permutation" and "eco.forestplot"
### (see ?eco.forestplot) when test = "bootstrap"

p <- plot(getis.ak)   ###  rankplot graph
p                     ###  points with colors of the color-scale:
                      ###  points with P < 0.05. Yellow points :
                      ###  points with P > 0.05
p <- plot(getis.ak, significant = FALSE)
p                     ###  all points have a color of the color-scale

### bootstrap example
getis.akb <- eco.lsa(eco[["P"]][, 1], con, method = "G*", nsim = 99,
test = "bootstrap")
p <- plot(getis.akb)     ### forestplot graph
p + ggplot2::theme_bw()  ### the plot can be modified with ggplot2
                         ### In this case, the background color is modified


## Getis-Ord's G

data(eco.test)
require(ggplot2)
con<- eco.weight(eco[["XY"]], method = "knearest", k = 4)
```

```
### self = FALSE for G
getis <- eco.lsa(eco[["P"]][, 1], con, method = "G", nsim = 99, adjust = "none")
plot(getis)


## Local Moran's I

 #-------------------------
 # TESTING PHENOTYPIC DATA-
 #-------------------------

 con <- eco.weight(eco[["XY"]], method = "knearest",  k = 4, row.sd = TRUE)
 # row standardized weights = TRUE

 # test for the first trait of the data frame P
 localmoran <- eco.lsa(eco[["P"]][, 1], con, method = "I", nsim = 99)

 plot(localmoran)

 # test for several variables

 all.traits <- apply(eco[["P"]], 2, eco.lsa,  con, method = "I", nsim = 99)

 # Observed statistic and P-values tables (individuals x traits)
 stat.P <- sapply(all.traits, function(x) return(ecoslot.OUT(x)[,1]))
 pval.P <- sapply(all.traits, function(x) return(ecoslot.OUT(x)[,4]))

 # Plot of the phenotypic spatial patterns

 par(mfrow = c(2,4))
 for(i in 1:8) {
 image(matrix(stat.P[,i], 15,15))
 }

 par(mfrow = c(2,4))
 for(i in 1:8) {
 image(matrix(pval.P[,i], 15,15))
 }


 #-------------------------
 # TESTING GENOTYPIC DATA-
 #-------------------------

 # eco[["A"]] is a matrix with the genetic data of "eco"
 # as frequencies for each allele in each individual.

 head(eco[["A"]])      # head of the matrix - 40 alleles

 con <- eco.weight(eco[["XY"]], method = "knearest",  k = 4, row.sd = TRUE)
 # row standardized weights = TRUE

 # test for a single allele
```

```
localmoran.geno <-  eco.lsa(eco[["A"]][, 32], con, method = "I", nsim = 99)

# test for several alleles -  40 alleles (it runs in less than 1 min
# for 99 simulations per allele;  999 simulations takes ~ 11 s per allele,
# less than 8 min in total.)
all.alleles <- apply(eco[["A"]], 2, eco.lsa,  con, method = "I", nsim = 99)

# plot all alleles to get an overview of the spatial patterns
lapply(all.alleles, plot)

# Observed statistic and P-values tables (individuals x loci)
stat.G <- sapply(all.alleles, function(x) return(ecoslot.OUT(x)[,1]))
pval.G <- sapply(all.alleles, function(x) return(ecoslot.OUT(x)[,4]))

# counting individuals with P < 0.05 for each allele
# (5 * 225 /100 ~  12   significant tests by random)
signif <- lapply(all.alleles, function(x) sum(ecoslot.OUT(x)[,4] < 0.05))
signif <- unlist(signif)

# filtering alleles, loci with > 12 significant individual tests

A.local <- eco[["A"]][, signif > 12]     #filtered matrix
stat.G.f <- stat.G[, signif > 12]
pval.G.f <- pval.G[, signif > 12]

# Plot of the genotypic spatial patterns

# one plot possibility, using the EcoGenetics method <rankplot>
all.local <- all.alleles[signif > 12]
lapply(all.local, plot)

# other plot possibility, using <image>
par(mfrow = c(3,4))
for(i in 1:12) {
image(matrix(stat.G[,i], 15,15))
}

par(mfrow = c(3,4))
for(i in 1:12) {
image(matrix(pval.G[,i], 15,15))
}


## Local Geary's C

data(eco.test)
require(ggplot2)
con<- eco.weight(eco[["XY"]], method = "knearest",  k = 4, row.sd = TRUE)
### row standardized weights = TRUE
localgeary <- eco.lsa(eco[["P"]][, 1], con, method = "C", nsim = 99,
adjust = "none")
plot(localgeary)
```

```
#---Moran's I, Geary's C and bivariate Moran's Ixy correlograms---#


## Moran's I correlogram

### single test
data(eco.test)
require(ggplot2)
moran <- eco.correlog(Z=eco[["P"]][,1], XY = eco[["XY"]], method = "I",
smax=10, size=1000)
plot(moran)

### multiple tests
moran2 <- eco.correlog(Z=eco[["P"]], XY = eco[["XY"]], method = "I",
smax=10, size=1000)
plot(moran2, var ="P2") ## single plots
plot(moran2, var ="P3") ## single plots

graf <- plot(moran2, meanplot = TRUE)  ## multiple plot with mean correlogram
                                       ## and jackknifed confidence intervals.

plot(graf[[1]])
plot(graf[[2]])

### correlogram plots support the use of ggplot2 syntax
moranplot <- plot(moran2, var ="P3") + theme_bw() + theme(legend.position="none")
moranplot


## Geary's C correlogram

data(eco.test)
require(ggplot2)
geary <- eco.correlog(Z = eco[["P"]][,1], XY = eco[["XY"]], method = "C",
smax=10, size=1000)
plot(geary)


## Moran's Ixy cross-correlogram

data(eco.test)
require(ggplot2)
cross <- eco.correlog(Z = eco[["P"]][,1], XY = eco[["XY"]], Y = eco[["E"]][, 1],
method = "CC", int = 2, smax = 15)
plot(cross)


#---Mantel and partial Mantel correlograms---#


data(eco.test)
require(ggplot2)
```

```
corm <- eco.cormantel(M = dist(eco[["P"]]), size=1000,smax=7, XY = eco[["XY"]],
nsim = 99)
plot(corm)

corm <- eco.cormantel(M = dist(eco[["P"]]), size=1000,smax=7, XY = eco[["XY"]],
nsim = 99, test = "bootstrap")
plot(corm)

## variogram plots support the

## partial Mantel correlogram

corm <- eco.cormantel(M = dist(eco[["P"]]), MC = dist(eco[["E"]]),
size=1000, smax=7, XY = eco[["XY"]], nsim = 99)
plot(corm)

### correlogram plots support the use of ggplot2 syntax
mantelplot <- plot(corm) + theme_bw() + theme(legend.position="none")
mantelplot


#---Empirical variogram---#


data(eco.test)
require(ggplot2)
variog <- eco.variogram(Z = eco[["P"]][, 2],XY =  eco[["XY"]])
plot(variog)
use of ggplot2 syntax
variogplot <- plot(variog) + theme_bw() + theme(legend.position="none")
variogplot


#---Computing NDVI with atmospheric correction
#---over a time series, and estimation of the temporal mean---#

require(raster)
data(tab)
data(eco3)
set.seed(6)

temp <- list()

## we create 4 simulated rasters for the data included in the object tab:

for(i in 1:4) {
temp[[i]] <- runif(19800, 0, 254)
temp[[i]] <- matrix(temp[[i]], 180, 110)
temp[[i]] <- raster(temp[[i]], crs="+proj=utm")
extent(temp[[i]])<-c(3770000, 3950000, 6810000, 6920000)
}

writeRaster(temp[[1]], "20040719b4.tif", overwrite = T)
```

```
writeRaster(temp[[2]], "20040719b3.tif", overwrite = T)
writeRaster(temp[[3]], "20091106b4.tif", overwrite = T)
writeRaster(temp[[4]], "20091106b3.tif", overwrite = T)

## Computing NDVI images:

eco.NDVI(tab, "COST", "NDVI", "LT5")

## Mean NDVI image computed over the NDVI images that we calculated:

eco.NDVI.post(tab, "COST", "NDVI", what = c("mean", "var"))
mean.ndvi <- raster("NDVI.COST.mean.tif")
plot(mean.ndvi)

## Extraction of the mean NDVI for each point in the object eco and plot of the data:

ndvi <- extract(mean.ndvi, eco3[["XY"]])
ndvi<- aue.rescale(ndvi)
plot(eco3[["XY"]][, 1], eco3[["XY"]][, 2], col=rgb(ndvi, 0, 0),
pch=15, main = "Mean NDVI", xlab = "X", ylab  = "Y")


#---Theil-sen estimation for a raster---#

require("raster")
set.seed(6)

temp <- list()

for(i in 1:100) {
temp[[i]] <- runif(36,-1, 1)
temp[[i]] <- matrix(temp[[i]], 6, 6)
temp[[i]] <- raster::raster(temp[[i]])
}

temp <- brick(temp)

writeRaster(temp,"temporal.tif", overwrite=T)
rm(temp)
ndvisim <- brick("temporal.tif")
date <- seq(from = 1990.1, length.out = 100, by = 0.2)
eco.theilsen(ndvisim, date)
pvalue <- raster("pvalue.tif")
slope <- raster("slope.tif")
par(mfrow = c(1, 2))
plot(pvalue, main = "p-value")
plot(slope, main = "slope")


#---Conversion of ecogen data to several formats and format tool---#


## Interconverting an ecogen genetic data frame among several formats
```

```
data(eco3)

### One allele per column
loc2al <- eco.convert(eco3[["G"]], "matrix", "alleles.matrix", ploidy = 2)
loc2al

### Inverse operation (collapse alleles into locus)
al2loc <- eco.convert(loc2al, "alleles.matrix", "matrix", ploidy = 2)
al2loc

### Separating alleles with a character string
loc2loc <- eco.convert(eco3[["G"]], "matrix", "matrix", ploidy = 2, sep.out = "/")
loc2loc

### Inverse operation (removing separator)
loc2loc.nosep <- eco.convert(loc2loc, "matrix", "matrix", ploidy = 2, sep.in = "/", sep.out = "")
loc2loc.nosep


## Format tool

data(eco.test)

### Adding zeros

example <- as.matrix(genotype[1:10,])
mode(example) <- "character"
### example data
example
recoded <- eco.format(example, ncod = 1, ploidy = 2, nout = 3)
### recoded data
recoded
### leading zeros
recoded2 <- eco.format(example, ncod = 1, ploidy = 2, nout = 3,
fill.mode = "first")
### recoded data
recoded2


### Tetraploid data, separating alleles with a "/"
tetrap <- as.matrix(example)
### simulated tetraploid example data
tetrap <- matrix(paste(example,example, sep = ""), ncol = ncol(example))
recoded <- eco.format(tetrap, ncod = 1, ploidy = 4, sep.out = "/")
### recoded data
recoded


## Creating input data for Geneland with an ecogen object

data(eco.test)
eco.2geneland(eco, 1)
```

```
## Exporting an ecogen genetic data frame into Genepop format

data(eco.test)
eco.2genepop(eco, grp = "pop", name = "infile.genepop.txt")
### an output file "infile.genepop.txt" is generated in the working directory


## Converting a diploid ecogen genetic data frame into a gstudio object

data(eco.test)
gsteco <- eco.2gstudio(eco, "separated")
gsteco


## Converting an ecogen genetic data frame into a hierfstat data frame

data(eco.test)
hiereco <- eco.2hierfstat(eco, "pop")


## Exporting an ecogen genetic data frame into SPAGeDI format

data(eco.test)
eco.2spagedi(eco, "pop", ndig = 1,int=2, smax=6, name="infile.spagedi.txt")


## End(Not run)
```

`aue.image2df` *Transforming a raster into a data frame with cartesian coordinates*

Description

This function returns a data frame with the column number (x), row number (y) and cell value (z) of each pixel in a raster.

Usage

```
aue.image2df(mat)
```

Arguments

`mat`	Input raster matrix.

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

Examples

```
## Not run:

ras <- matrix(eco[["P"]][,1],15,15)
image(ras)
ras.row <- aue.image2df(ras)
ras.row
image(matrix(ras.row[,3], 15, 15))


## End(Not run)
```

`aue.rescale` *Scaling a data frame or matrix to 0 - 1 range*

Description

This program scales each column of a data frame or a matrix to 0-1 range, computing ((X)*ij* - (Xmin)*i*) / range((X)*i*) for each individual *j* of the variable *i*.

Usage

```
aue.rescale(dfm)
```

Arguments

`dfm`	Data frame, matrix or vector to scale.

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

Examples

```
## Not run:

data(eco.test)
require(adegenet)
pc <- dudi.pca(eco[["P"]], scannf = FALSE, nf = 3)
pc <- pc$li
pc <- aue.rescale(pc)
plot(eco[["XY"]][, 1], eco[["XY"]][, 2], col = rgb(pc), pch = 16,
cex = 1.5, xlab ="X", ylab= "Y")


## End(Not run)
```

aue.sort *Ordering the content of cells in a matrix. Ordering alleles in a genetic matrix.*

Description

This program takes a matrix and orders the content of each cell. It was specially designed for genetic data, but can be used with any data that can be rearrenged by the function order. The arguments ploidy and ncode determine the mode of ordering the data. The cells corresponding to each individual *i* and loci *j* are ordered in ascending order in default option (it can be passed decreasing = TRUE as argument, if descending order is desired). For example, a locus with ploidy = 2 and ncod =1, coded as 51, for an individual, will be recoded as 15. A locus with ploidy = 3 and coded as 143645453, will be recoded as 143453645 (alleles 143, 454 and 645).

Usage

```
aue.sort(X, ncod = NULL, ploidy = 2, sep.loc = "", chk.plocod = TRUE,
  ...)
```

Arguments

X	Any matrix with content to order.
ncod	Number of digits coding each allele (e.g., 1: x, 2: xx, 3: xxx, etc.). If NULL, ncode will we obtained from the ploidy and the maximum number of characters in the data cells.
ploidy	Ploidy of the data.
sep.loc	Character string separating alleles.
chk.plocod	Defalult TRUE. The function checks coherence in ploidy and number of digits coding alleles.
...	Additional arguments passed to order

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

Examples

```
## Not run:

# Example 1----------------------

geno <- c(12, 52, 62, 45, 54, 21)
geno <- matrix(geno, 3, 2)

# ordering the data
aue.sort(geno, ploidy = 2)
```

```
# decreasing sort order
aue.sort(geno, ploidy = 2, decreasing = TRUE)


# Example 2----------------------

geno2 <- c(123456, 524556, 629359, 459459, 543950, 219405)
geno2 <- matrix(geno2, 3, 2)

# ordering the data as diploid
aue.sort(geno2, ploidy = 2)  # the data is ordered using blocks of 3 characters

# ordering the data as triploid
aue.sort(geno2, ploidy = 3)  # the data is ordered using blocks of 2 characters

# error: the ploidy and the number of characters are not congruent
aue.sort(geno2, ploidy = 5)

# error: the ploidy and the number of characters are not congruent
aue.sort(geno2, ploidy = 5)


# Example 3----------------------

# any character data
generic <- c("aldk", "kdbf", "ndnd", "ndkd")
generic <- matrix(generic, 2, 2)
aue.sort(generic, ploidy = 2)
aue.sort(generic, ploidy = 4)


## End(Not run)
```

`coordinates` *Coordinates*

Description

Data frame with cartesian coordinates of 225 simulated individuals.

Usage

```
data(eco.test)
coordinates
```

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

eco *Eco*

Description

ecogen object with simulated data of 225 individuals.

Usage

```
data(eco.test)
eco
```

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

eco.2geneland *Creating input data for Geneland with an ecogen object*

Description

This function creates four data frames in the working directory (XY.txt, NAMES.txt, P.txt, G.txt) which can be loaded in Geneland.

Usage

```
eco.2geneland(eco, ncod = NULL, ploidy = 2)
```

Arguments

eco	Object of class "ecogen".
ncod	Number of digits coding each allele (e.g., 1: x, 2: xx, 3: xxx, etc.).
ploidy	Ploidy of the data.

Value

XY.txt Matrix with coordinates.

NAMES.txt Matrix with row names.

P.txt Matrix with phenotypic data.

G.txt Matrix with genotypic data.

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

Examples

```
## Not run:

data(eco.test)
eco.2geneland(eco, 1)


## End(Not run)
```

`eco.2genepop` *Exporting an ecogen genetic data frame into Genepop format*

Description

This function converts the genetic data of an ecogen object into a Genepop input file.

Usage

```
eco.2genepop(eco, name = "infile.genepop.txt", grp = NULL, nout = 3,
  sep = "")
```

Arguments

`eco`	Object of class "ecogen".
`name`	The name of the output file.
`grp`	The name of the S slot column with groups in which the sample must be divided (e.g., populations). If groups are not given (grp = NULL), all individuals will be assigned to a single one.
`nout`	Number of digits in the output file
`sep`	Character separating alleles.

Value

A Genepop file in the working directory.

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

Examples

```
## Not run:

data(eco.test)
eco.2genepop(eco, grp = "pop", name = "infile.genepop.txt")
# an output file "infile.genepop.txt" is generated in the working directory


## End(Not run)
```

eco.2gstudio *Converting a diploid ecogen genetic data frame into a gstudio object*

Description

This function converts the genetic data of an ecogen object in a gstudio data frame.

Usage

```
eco.2gstudio(eco, type = "separated", ...)
```

Arguments

eco	Object of class "ecogen".
type	The type of data passed to gstudio locus. Default is "separated" (data as microsatellites or individual haplotypes); "aflp" for presence - absence data.
...	Further arguments passed to df2genind.

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

Examples

```
## Not run:

data(eco.test)
gsteco <- eco.2gstudio(eco, "separated")
gsteco


## End(Not run)
```

eco.2hierfstat *Converting an ecogen genetic data frame into a hierfstat data frame*

Description

This function converts the genetic data of an ecogen object in a hierfstat data frame.

Usage

```
eco.2hierfstat(eco, pop = NULL)
```

Arguments

eco	Object of class "ecogen".
pop	The name of the S slot column with the groups for the hierfstat data frame.

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

Examples

```
## Not run:

data(eco.test)
hiereco <- eco.2hierfstat(eco, "pop")
require("hierfstat")
basic.stats(hiereco)


## End(Not run)
```

---

eco.2spagedi *Exporting an ecogen genetic data frame into SPAGeDI format*

Description

This function converts the genetic data of an ecogen object in a SPAGeDI input file. When distance classes are required, they can be constructed by combining the parameters "int", "smin", "smax", "nclass", "seqvec" and "size", as described in the function eco.lagweight. A distance matrix can also be included using the "distmat" parameter. Missing data must be coded as a single "NA" in the G data frame.

Usage

```
eco.2spagedi(eco, pop = NULL, ndig, name = "infile.spagedi.txt", smin = 0,
  smax = NULL, int = NULL, nclass = NULL, seqvec = NULL, size = NULL,
  bin = c("sturges", "FD"), distmat = NULL, latlon = FALSE)
```

Arguments

eco	Object of class "ecogen".
pop	The name of the S slot column with the groups for the output data. The default option includes all the individuals into a single group.
ndig	Number of digits coding each allele (e.g., 1: x, 2: xx, or 3: xxx).
name	The name of the output file.
smin	Minimum class distance in the units of the XY slot data.
smax	Maximum class distance in the units of the XY slot data.

int	Distance interval in the units of the XY slot data.
nclass	Number of classes.
seqvec	Vector with breaks in the units of the XY slot data.
size	Number of individuals per class.
bin	Rule for constructing intervals when a partition parameter (int, nclass or size) is not given. Default is Sturge's rule (Sturges, 1926). Other option is Freedman-Diaconis method (Freedman and Diaconis, 1981).
distmat	Distance matrix to include (optional).
latlon	Are the coordinates in decimal degrees format? Defalut FALSE. If TRUE, the coordinates must be in a matrix/data frame with the longitude in the first column and latitude in the second. The position is projected onto a plane in meters with the function geoXY.

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

References

Freedman D., and P. Diaconis. 1981. On the histogram as a density estimator: L 2 theory. Probability theory and related fields, 57: 453-476.

Hardy O. and X Vekemans. 2002. SPAGeDi: a versatile computer program to analyse spatial genetic structure at the individual or population levels. Molecular ecology notes, 2: 18-620.

Sturges H. 1926. The choice of a class interval. Journal of the American Statistical Association, 21: 65-66.

Examples

```
## Not run:

data(eco.test)
eco.2spagedi(eco, "pop", ndig = 1,int=2, smax=6, name="infile.spagedi.txt")


## End(Not run)
```

---

eco.alfreq *Allelic frequency histograms for an ecogen genetic data frame*

Description

This program computes the frequency of each allele and plots a histogram for the number of alleles with a given frequency. The distribution is expected to be L-shaped under mutation-drift equilibrium. When a factor is given, the program plots a histogram for each group.

Usage

```
eco.alfreq(eco, grp = NULL)
```

Arguments

eco	Object of class "ecogen".
grp	Optional factor (column of the S slot) for plots by group.

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

References

Luikart G., F. Allendorf, F. Cornuet, and W. Sherwin. 1998. Distortion of allele frequency distributions provides a test for recent population bottlenecks. Journal of Heredity, 89: 238-247.

Examples

```
## Not run:

data(eco.test)
eco.alfreq(eco)
eco.alfreq(eco, "pop")


## End(Not run)
```

---

eco.association *Chi-square and Fisher's exact test for association of loci and alleles with a factor*

Description

Chi-square and Fisher's exact test for association of loci and alleles with a factor

Usage

```
eco.association(eco, assoc = c("within", "between"), x,
  method = c("fisher.test", "chisq.test"), nrep = 99, adjust = "none",
  ndig = NA)
```

Arguments

eco	Object of class "ecogen".
assoc	"between" if the association test should be performed between a factor and a loci, or "within" if the association test should be performed between a factor and alleles within loci. For haploid data, use option "within".
x	The name of the S slot column with the groups for the association test.
method	Test method ("chisq.test" or "fisher.test"). Default is "fisher.test".
nrep	Number of repetitions for the permutation test.
adjust	Correction method of P-values for multiple tests, passed to p.adjust. Default is "none" (no correction).
ndig	Number of digits coding each alleles (e.g. 2: xx, or 3: xxx) when assoc is "within".

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

See Also

chisq.test fisher.test fisher.test

Examples

```
## Not run:

data(eco.test)
eco.association(eco, "within", "pop")
eco.association(eco, "within", "pop", adjust="fdr")
eco.association(eco, "within", "pop", method = "chisq.test")
eco.association(eco, "between", "pop", ndig = 1)
eco.association(eco, "between", "pop", method = "chisq.test", ndig = 1)


## End(Not run)
```

---

eco.cbind *Combining the columns of ecogen object*

Description

Combining the columns of ecogen object

Usage

```
eco.cbind(eco1, eco2, ..., missing = c("0", "MEAN", "NA"))
```

Arguments

eco1	Object of class "ecogen".
eco2	Object of class "ecogen".
...	Other "ecogen" objects to combine and the specification of the data frames to combine. Can be any of the following(s): P","G", "E", "S", "C", or "ALL" (default). If a "G" data frame is provided, the program also generates the A slot coding the missing data as "0" in default option (see the argument "missing"). The XY slot is generated automatically if present.
missing	Missing data manipulation. It can take three values ("0" ,"NA" or "MEAN"- i.e, the mean frequency of the corresponding allele). Missing elements are coded as 0 in the default option.

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

Examples

```
## Not run:

data(eco.test)
eco.example <- eco.cbind(eco,eco,"ALL")
eco.example
eco.example2 <- eco.cbind(eco, eco,"P", "G", missing="NA")
eco.example2


## End(Not run)
```

---

eco.clear *Clearing the working environment, maintaining only the specified objects*

Description

This function removes all the elements of the working environment, with the exception of those included in the argument of the function. Hidden elements can also be removed by setting all = TRUE.

Usage

```
eco.clear(..., all = FALSE)
```

Arguments

...	Objects to retain.
all	Remove also hidden elements? Default FALSE.

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

Examples

```
## Not run:

data(eco.test)
ls()
eco.clear(eco)
ls()


## End(Not run)
```

---

eco.convert *Conversion utility for genetic data*

Description

This function interconverts genetic data among matrix format (one locus or one allele per column) and list format (one locus or one allele per column).

Usage

```
eco.convert(X, input = c("matrix", "alleles.matrix", "list", "alleles.list"),
  output = c("matrix", "alleles.matrix", "list", "alleles.list"),
  ncod = NULL, ploidy = 2, sep.in, sep.out, chk.names = TRUE,
  chk.plocod = TRUE)
```

Arguments

X	Input data.
input	Input data format.
output	Output data format.
ncod	Number of digits coding each allele.
ploidy	Ploidy of the data.
sep.in	Character separating alleles in the input data if present.
sep.out	Character separating alleles in the output data. Default option do not separate alleles.
chk.names	Defalult TRUE. The function makes checks of individuals and loci names during conversion.
chk.plocod	Defalult TRUE. The function checks coherence in/between ploidy and number of digits coding alleles for loci data during conversion.

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

Examples

```
## Not run:

data(eco3)

# One allele per column
loc2al <- eco.convert(eco3[["G"]], "matrix", "alleles.matrix", ploidy = 2)
loc2al

# Inverse operation (collapse alleles into locus)
al2loc <- eco.convert(loc2al, "alleles.matrix", "matrix", ploidy = 2)
al2loc

# Separating alleles with a character string
loc2loc <- eco.convert(eco3[["G"]], "matrix", "matrix", ploidy = 2, sep.out = "/")
loc2loc

# Inverse operation (removing separator)
loc2loc.nosep <- eco.convert(loc2loc, "matrix", "matrix", ploidy = 2, sep.in = "/", sep.out = "")
loc2loc.nosep

# Locus to list
loc2list <- eco.convert(eco3[["G"]], "matrix", "list", ploidy = 2)
loc2list

# Locus to allele list
al2list <- eco.convert(eco3[["G"]], "matrix",  "alleles.list", ploidy = 2)
al2list

# The inverse operations are also defined. All the formats are interconvertible.
# Locus operations have defined a within operation (matrix to matrix, list to list),
# with the purpose of put/remove separators between alleles. The program accepts any ploidy level.



## End(Not run)
```

---

eco.cormantel *Mantel and partial Mantel correlograms*

Description

This program computes a Mantel correlogram for the data M, or a partial Mantel correlogram for the data M conditioned on MC, with P-values or bootstrap confidence intervals.

Usage

```
eco.cormantel(M, XY, MC = NULL, int = NULL, smin = 0, smax = NULL,
  nclass = NULL, seqvec = NULL, size = NULL, bin = c("sturges", "FD"),
  nsim = 99, classM = c("dist", "simil"), method = c("pearson",
  "spearman", "kendall"), test = c("permutation", "bootstrap"),
  alternative = c("auto", "two.sided", "greater", "less"), adjust = "holm",
  sequential = TRUE, latlon = FALSE, ...)
```

Arguments

M	Distance or similarity matrix.
XY	Data frame or matrix with individual's positions (projected coordinates).
MC	Distance or similarity matrix (optional).
int	Distance interval in the units of XY.
smin	Minimum class distance in the units of XY.
smax	Maximum class distance in the units of XY.
nclass	Number of classes.
seqvec	Vector with breaks in the units of XY.
size	Number of individuals per class.
bin	Rule for constructing intervals when a partition parameter (int, nclass or size) is not given. Default is Sturge's rule (Sturges, 1926). Other option is Freedman-Diaconis method (Freedman and Diaconis, 1981).
nsim	Number of Monte-Carlo simulations.
classM	Are M and MC distance or similarity matrices? Default option is classM = "dist" (distance). For similarity, classM = "simil". An incorrect option selected will generate an inverted plot.
method	Correlation method used for the construction of the statistic ("pearson", "spearman" or "kendall"). Kendall's tau computation is slow.
test	If test = "bootstrap", the program generates a bootstrap resampling and the associated confidence intervals. If test = "permutation" (default) a permutation test is made and the P-values are computed.
alternative	The alternative hypothesis. If "auto" is selected (default) the program determines the alternative hypothesis. Other options are: "two.sided", "greater" and "less".
adjust	Correction method of P-values for multiple tests, passed to p.adjust. Defalut is "holm".
sequential	Should be performed a Holm-Bonberroni (Legendre and Legendre, 2012) adjustment of P-values? Defalult TRUE.
latlon	Are the coordinates in decimal degrees format? Defalut FALSE. If TRUE, the coordinates must be in a matrix/data frame with the longitude in the first column and latitude in the second. The position is projected onto a plane in meters with the function geoXY.
...	Additional arguments passed to cor.

Value

The program returns an object of class "eco.correlog" with the following slots:

> OUT analysis output

> IN input data of the analysis

> BEAKS breaks

> CARDINAL number of elements in each class

> NAMES variables names

> METHOD analysis method

> DISTMETHOD method used in the construction of breaks

> TEST test method used (bootstrap, permutation)

> NSIM number of simulations

> PADJUST P-values adjust method for permutation tests

ACCESS TO THE SLOTS The content of the slots can be accessed with the corresponding accessors, using the generic notation of EcoGenetics (<ecoslot.> + <name of the slot> + <name of the object>). See help("EcoGenetics accessors") and the Examples section below.

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

References

Freedman D., and P. Diaconis. 1981. On the histogram as a density estimator: L 2 theory. Probability theory and related fields, 57: 453-476.

Legendre P., and L. Legendre. 2012. Numerical ecology. Third English edition. Elsevier Science, Amsterdam, Netherlands.

Oden N., and R. Sokal. 1986. Directional autocorrelation: an extension of spatial correlograms to two dimensions. Systematic Zoology, 35:608-617

Sokal R. 1986. Spatial data analysis and historical processes. In: E. Diday, Y. Escoufier, L. Lebart, J. Pages, Y. Schektman, and R. Tomassone, editors. Data analysis and informatics, IV. North-Holland, Amsterdam, The Netherlands, pp. 29-43.

Sturges H. 1926. The choice of a class interval. Journal of the American Statistical Association, 21: 65-66.

Examples

```
## Not run:

data(eco.test)
require(ggplot2)

corm <- eco.cormantel(M = dist(eco[["P"]]), size=1000,smax=7, XY = eco[["XY"]],
nsim = 99)
plot(corm)
```

```
corm <- eco.cormantel(M = dist(eco[["P"]]), size=1000,smax=7, XY = eco[["XY"]],
nsim = 99, test = "bootstrap")
plot(corm)

# partial Mantel correlogram
corm <- eco.cormantel(M = dist(eco[["P"]]), MC = dist(eco[["E"]]),
size=1000, smax=7, XY = eco[["XY"]], nsim = 99)
plot(corm)

# correlogram plots support the use of ggplot2 syntax
mantelplot <- plot(corm) + theme_bw() + theme(legend.position="none")
mantelplot


#-----------------------
# ACCESSORS USE EXAMPLE
#-----------------------

# the slots are accesed with the generic format
# (ecoslot. + name of the slot + name of the object).
# See help("EcoGenetics accessors")

ecoslot.OUT(corm)        # slot OUT
ecoslot.BREAKS(corm)     # slot BREAKS


## End(Not run)
```

`eco.correlog` *Moran's I, Geary's C and bivariate Moran's I correlograms*

Description

This program computes Moran's, Geary's and bivariate Moran's correlograms, for single or multiple variables, with P-values or bootstrap confidence intervals. The program allows high flexibility for the construction of intervals. For detailed information about the range partition methods see `eco.lagweight`

Usage

```
eco.correlog(Z, XY, Y = NULL, int = NULL, smin = 0, smax = NULL,
  nclass = NULL, size = NULL, seqvec = NULL, method = c("I", "C", "CC"),
  nsim = 99, test = c("permutation", "bootstrap"), alternative = c("auto",
  "two.sided", "greater", "less"), adjust = "holm", sequential = TRUE,
  include.zero = TRUE, cummulative = FALSE, bin = c("sturges", "FD"),
  row.sd = FALSE, latlon = FALSE)
```

Arguments

Z	Vector, matrix or data frame with variable/s (in matrix or data frame formats, variables in columns).
XY	Data frame or matrix with individual's positions (projected coordinates).
Y	Vector with the second variable for Mantel's Ixy cross-correlograms. If Z has multiple variables, the program will compute the cross-correlograms for each with Y.
int	Distance interval in the units of XY.
smin	Minimum class distance in the units of XY.
smax	Maximum class distance in the units of XY.
nclass	Number of classes.
size	Number of individuals per class.
seqvec	Vector with breaks in the units of XY.
method	Correlogram method. Could be I for Moran's I, C for Geary's C and CC for Bivariate Moran's Ixy. If method = "CC", the program computes for the first interval (d = 0) the corresponding P-value and CI with cor.test.
nsim	Number of Monte-Carlo simulations.
test	If test = "bootstrap", the program generates a bootstrap resampling and the associated confidence intervals of the null hypothesis. If test = "permutation" (default) a permutation test is made and the P-values are computed.
alternative	The alternative hypothesis. If "auto" is selected (default) the program determines the alternative hypothesis. Other options are: "two.sided", "greater" and "less".
adjust	P-values correction method for multiple tests passed to p.adjust. Defalut is "holm".
sequential	Should be performed a Holm-Bonberroni (Legendre and Legendre, 2012) adjustment of P-values? Defalult TRUE.
include.zero	Should be included the distance = 0 in cross correlograms (i.e., the intra- individual correlation)?. Defalut TRUE.
cummulative	Should be construced a cummulative correlogram?.
bin	Rule for constructing intervals when a partition parameter (int, nclass or size) is not given. Default is Sturge's rule (Sturges, 1926). Other option is Freedman-Diaconis method (Freedman and Diaconis, 1981).
row.sd	Logical. Should be row standardized the matrix? Default FALSE (binary weights).
latlon	Are the coordinates in decimal degrees format? Defalut FALSE. If TRUE, the coordinates must be in a matrix/data frame with the longitude in the first column and latitude in the second. The position is projected onto a plane in meters with the function geoXY.

Value

The program returns an object of class "eco.correlog" with the following slots:

> OUT analysis output

> IN analysis input data

> BEAKS breaks

> CARDINAL number of elements in each class

> NAMES variables names

> METHOD analysis method

> DISTMETHOD method used in the construction of breaks

> TEST test method used (bootstrap, permutation)

> NSIM number of simulations

> PADJUST P-values adjust method for permutation tests

ACCESS TO THE SLOTS The content of the slots can be accessed with the corresponding accessors, using the generic notation of EcoGenetics (<ecoslot.> + <name of the slot> + <name of the object>). See help("EcoGenetics accessors") and the Examples section below.

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

References

Freedman D., and P. Diaconis. 1981. On the histogram as a density estimator: L 2 theory. Probability theory and related fields, 57: 453-476.

Geary R. 1954. The contiguity ratio and statistical mapping. The incorporated statistician, 115-146.

Legendre P., and L. Legendre. 2012. Numerical ecology. Third English edition. Elsevier Science, Amsterdam, Netherlands

Moran P. 1950. Notes on continuous stochastic phenomena. Biometrika, 17-23.

Reich R., R. Czaplewski and W. Bechtold. 1994. Spatial cross-correlation of undisturbed, natural shortleaf pine stands in northern Georgia. Environmental and Ecological Statistics, 1: 201-217.

Sokal R. and N. Oden 1978. Spatial autocorrelation in biology: 1. Methodology. Biological journal of the Linnean Society, 10: 199-228.

Sokal R. and N. Oden. 1978. Spatial autocorrelation in biology. 2. Some biological implications and four applications of evolutionary and ecological interest. Biological Journal of the Linnean Society, 10: 229-49.

Sokal R. 1979. Ecological parameters inferred from spatial correlograms. In: G. Patil and M. Rosenzweig, editors. Contemporary Quantitative Ecology and elated Ecometrics. International Co-operative Publishing House: Fairland, MD, pp. 167-96.

Sturges H. 1926. The choice of a class interval. Journal of the American Statistical Association, 21: 65-66.

Examples

```
## Not run:

data(eco.test)
require(ggplot2)
```

```
##########################
# Moran's I correlogram
##########################

## single test with phenotypic traits
moran <- eco.correlog(Z=eco[["P"]][,1], XY = eco[["XY"]], method = "I", smax=10, size=1000)
plot(moran)

## multiple tests with phenotypic traits
moran2 <- eco.correlog(Z=eco[["P"]], XY = eco[["XY"]], method = "I", smax=10, size=1000)

plot(moran2, var ="P2") ## single plots
plot(moran2, var ="P3") ## single plots

graf <- plot(moran2, meanplot = TRUE)            ## multiple plot with mean correlogram
                                                 ## and jackknifed confidence intervals.

 plot(graf[[1]])
 plot(graf[[2]])

# correlogram plots support the use of ggplot2 syntax
moranplot <- plot(moran2, var ="P3") + theme_bw() + theme(legend.position="none")
moranplot

moranplot2 <- graf[[2]] + theme_bw() + theme(legend.position="none")
moranplot2


# single test with genotypic traits

# eco[["A"]] is a matrix with the genetic data of "eco"
# as frequencies for each allele in each individual. Each allele
# can be analyzed as single traits.

head(eco[["A"]])      # head of the matrix

# analyzing allele 1
moran <- eco.correlog(Z=[["A"]][,1], XY = eco[["XY"]], method = "I", smax=10, size=1000)
plot(moran)


# multiple tests with genotypic traits.
# nsim is set to 10 only for speed in the example
moran2 <- eco.correlog(Z = eco[["A"]], XY = eco[["XY"]], method = "I",smax=10, size=1000, nsim=99)


graf <- plot(moran2, meanplot = TRUE)              ## multiple plot with mean
                                                   ## correlogram and jackknifed
                                                   ## confidence intervals.

## the same example, but with nsim = 99.
```

```
moran3 <- eco.correlog(Z = eco[["A"]], XY = eco[["XY"]], method = "I", smax=10, size=1000, nsim=99)


plot(moran3, meanplot = TRUE, significant = TRUE)  ## plot for alleles with at least
                                                   ## one significant value after
                                                ## Bonferroni-Holm sequential P correction
                                                  ## (set adjust "none" for no family-wise
                                                     ## P correction in "eco.correlog")
#-----------------------
# ACCESSORS USE EXAMPLE
#-----------------------

# the slots are accesed with the generic format
# (ecoslot. + name of the slot + name of the object).
# See help("EcoGenetics accessors")

ecoslot.OUT(moran)      # slot OUT
ecoslot.BREAKS(moran)   # slot BREAKS

#-------------------------------------------------------------------------------#

############################
# Geary's C correlogram
############################

geary <- eco.correlog(Z = eco[["P"]][,1], XY = eco[["XY"]], method = "C",
smax=10, size=1000)
plot(geary)

#-------------------------------------------------------------------------------#

############################
# Bivariate Moran's Ixy
############################

cross <- eco.correlog(Z=eco[["P"]][,1], XY = eco[["XY"]], Y = eco[["P"]][, 1],
method = "CC", int= 2, smax=15)
plot(cross)


## End(Not run)
```

`eco.detrend` *Detrending spatial data with polynomial interpolation*

Description

This program performs a Trend Surface Analysis (Borcard et al. 2011, Legendre and Legendre 2012, Lichstein et al 2002) for the data Z and the given coordinates, projected or in decimal degrees format, in which case will be projected with geoXY.

Usage

```
eco.detrend(Z, XY, degree, center = TRUE, scale = FALSE, raw = FALSE,
  latlon = FALSE)
```

Arguments

Z	Data frame, matrix or vector with dependent variables.
XY	Data frame, matrix or vector with projected coordinates (X, XY or XYZ). For longitude-latitude data in decimal degrees format, use the option latlon = TRUE.
degree	Polynomial degree.
center	Should the data be centered? Default TRUE
scale	Should the data be scaled? Default FALSE
raw	Use raw and not orthogonal polynomials? Default FALSE
latlon	Are the coordinates in decimal degrees format? Defalut FALSE. If TRUE, the coordinates must be in a data.frame/matrix with the longitude in the first column and latitude in the second. The position is projected onto a plane in meters with the function geoXY.

Value

An object of class "eco.detrend" with the following slots:

> POLY.DEG polynomial degree used in the analysis

> RES detrended data

> XY projected coordinates

> MODEL models selected with the Akaike criterion

> ANALYSIS object of class "eco.mlm" with the regression results for each variable

ACCESS TO THE SLOTS The content of the slots can be accessed with the corresponding accessors, using the generic notation of EcoGenetics (<ecoslot.> + <name of the slot> + <name of the object>). See help("EcoGenetics accessors") and the Examples section below.

References

Borcard D., F. Gillet, and P. Legendre. 2011. Numerical ecology with R. Springer Science & Business Media.

Legendre P., and L. Legendre. 2012. Numerical ecology. Third English edition. Elsevier Science, Amsterdam, Netherlands.

Lichstein J., T. Simons, S. Shriner, and K. Franzreb. 2002. Spatial autocorrelation and autoregressive models in ecology. Ecological monographs, 72: 445-463.

Examples

```
## Not run:

data(eco2)

# original data
data1 <- matrix(eco2[["P"]][,1], 30, 30)
image(data1)

# original data + trend
data2 <- matrix(eco2[["P"]][,2], 30, 30)
image(data2)

# data detrending
data2.det <- eco.detrend(Z = eco2[["P"]][,2], XY =  eco2[["XY"]], degree =  1)


#-----------------------
# ACCESSORS USE EXAMPLE
#-----------------------

# the slots are accesed with the generic format
# (ecoslot. + name of the slot + name of the object).
# See help("EcoGenetics accessors")

data2.det <- ecoslot.RES(data2.det)       # detrended data in slot RES

data2.det <- matrix(data2.det[,1], 30, 30)
image(data2.det)



## End(Not run)
```

`eco.forestplot` *Forestplot graphs*

Description

This program generates a forest plot for the confidence interval of each individual of the input data (as row number) and the corresponding observed value of the statistic.

Usage

```
eco.forestplot(input, xlabel = NULL, ylabel = NULL, titlelabel = NULL,
  legendlabel = NULL)

## S4 method for signature 'eco.lsa'
eco.forestplot(input, xlabel = NULL, ylabel = NULL,
```

```
  titlelabel = NULL, legendlabel = NULL)

## S4 method for signature 'dataframeORmatrix'
eco.forestplot(input, xlabel = NULL,
  ylabel = NULL, titlelabel = NULL, legendlabel = NULL)
```

Arguments

input	Matrix/data frame, with three columns in the folowing order: observed value, lower and upper values of the confidence interval.
xlabel	Optional label for x axis.
ylabel	Optional label for y axis.
titlelabel	Optional title label.
legendlabel	Optional legend label.

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

Examples

```
## Not run:

require(ggplot2)
# simulated confidence intervals for the null hypotesis of a variable "a"
set.seed(8)
a<-runif(10, -2, 2)
infer <- runif(10, -1, 1)
super <- runif(10, -1, 1)
infer2 <- pmin(infer, super)
super2 <- pmax(infer, super)
data <- data.frame(a, infer2, super2)
forest <- eco.forestplot(data)
forest

# the forestplot method support the use of ggplot2 syntax
forest <- forest + theme_bw() + theme(legend.position="none")
forest

## End(Not run)
```

---

eco.format *Format tool for genetic data*

Description

Format tool for genetic data

Usage

```
eco.format(data, ncod = NULL, nout = 3, ploidy = 2, sep.in, sep.out,
  fill.mode = c("last", "first", "none"), recode = c("none", "all",
  "column"), show.codes = FALSE)
```

Arguments

data	Genetic data frame.
ncod	Number of digits coding each allele in the input file.
nout	Number of digits in the output.
ploidy	Ploidy of the data.
sep.in	Character separating alleles in the input data if present.
sep.out	Character separating alleles in the output data. Default
fill.mode	Add zeros at the beggining ("fist") or the end ("last") of each allele. Default = "last".
recode	Recode mode: "none" for no recoding (defalut), "all" for recoding the data considering all the individuals values at once (e.g., protein data), or "column" for recoding the values by column (e.g., microsatellite data).
show.codes	May we returned tables with the equivalence between the old and new codes when recode = "all" or recode = "column"?

Details

The function can format data with different ploidy levels. It allows to: - add/remove zeros at the beginning/end of each allele - separate alleles with a character - divide alleles into columns - bind alleles from separate columns - transform character data into numeric data

"NA" is considered special character (not available data).

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

Examples

```
## Not run:

data(eco.test)

# Adding zeros

example <- as.matrix(genotype[1:10,])
mode(example) <- "character"
# example data
example
recoded <- eco.format(example, ncod = 1, ploidy = 2, nout = 3)
# recoded data
recoded
```

```
# Tetraploid data, separating alleles with a "/"
tetrap <- as.matrix(example)
# simulated tetraploid example data
tetrap <- matrix(paste(example,example, sep = ""), ncol = ncol(example))
recoded <- eco.format(tetrap, ncod = 1, ploidy = 4, sep.out = "/")
# recoded data
recoded

# Example with a single character
ex <- c("A","T","G","C")
ex <- sample(ex, 100, rep= T)
ex <- matrix(ex, 10, 10)
colnames(ex) <- letters[1:10]
rownames(ex) <- LETTERS[1:10]
# example data
ex
recoded <- eco.format(ex, ploidy = 1, nout = 1,  recode = "all", show.codes = TRUE)
# recoded data
recoded


# Example with two strings per cell and missing values:
ex <- c("Ala", "Asx", "Cys", "Asp", "Glu", "Phe", "Gly", "His", "Ile",
"Lys", "Leu", "Met", "Asn", "Pro", "Gln", "Arg", "Ser", "Thr",
"Val", "Trp")
ex1 <- sample(ex, 100, rep= T)
ex2 <- sample(ex, 100, rep= T)
ex3 <- paste(ex1, ex2, sep="")
missing.ex3 <- sample(1:100, 20)
ex3[missing.ex3] <-NA
ex4 <- matrix(ex3, 10, 10)
colnames(ex4) <- letters[1:10]
rownames(ex4) <- LETTERS[1:10]
# example data
ex4
recoded <- eco.format(ex4, ncod = 3, ploidy = 2,
                      nout = 2, recode = "column")
# recoded data
recoded

# Example with a vector, following the latter example:
ex1 <- as.data.frame(ex1)
# example data
ex1
recoded <- eco.format(ex1, ploidy = 1, recode = "all")
# recoded data
recoded


## End(Not run)
```

eco.genepop2df *Importing a Genepop file*

Description

This function converts a Genepop file into an object with a genetic matrix (G) and a structures matrix (S).

Usage

```
eco.genepop2df(genefile = NULL)
```

Arguments

genefile	Genepop file.

Value

A list with the objects G (genetic matrix) and S (structures matrix).

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>, adapting code written by Emiel van Loon and Scott Davis

Examples

```
## Not run:
# ingpop, file with Genepop format in the folder "/extdata" of the package

ecopath <- paste(path.package("EcoGenetics"), "/extdata/ingpop", sep = "")
ingpop <- eco.genepop2df(ecopath)
ingpop

## End(Not run)
```

eco.gsa *Global spatial analysis*

Description

This program computes Moran's I, Geary's C, Bivariate Moran's I or Join-count statistics with P-values.

The program allows the analysis of a single variable or multiple variables. in the last case, the variables must be in columns and the individuals in rows.

For join-count analysis, a ncod argument must be supplied, and in the case of genetic data, an additional ploidy argument. The data is then ordered with the function aue.sort. This step is required in the analysis of genotypes. An individual with the alleles A and B, coded as AB, is identical to other coded as BA. The ordination step ensures that both are considered in a single category. For the analysis of frequencies of single alleles, the input is count data (ploidy-times the frequency, as provided by the slot A of an ecogen object: the count data A' can be obtained as A' <- ploidy * A), using the function with the arguments ploidy = 1 and ncod = 1.

Usage

```
eco.gsa(Z, Y = NULL, con, method = c("I", "C", "CC", "JC"), ncod = NULL,
  ploidy = 1, nsim = 99, alternative = c("auto", "two.sided", "greater",
  "less"), adjust = "fdr", row.sd = FALSE, plotit = TRUE)
```

Arguments

Z	Vector with a variable, or matrix/data frame with variables in columns.
Y	Vector with the second variable for Moran's Ixy. If Z has multiple variables, the program will compute the coefficent for each with Y.
con	An object of class eco.weight obtained with the function eco.weight, a listw object or a matrix, giving the spatial weights for the analysis. If "con" is a matrix, an attribute "xy" including the projected coordinates is required.
method	Method of analysis: "I" for Moran's I, "C" for Geary's C, "CC" for the Bivariate Moran's or "JC" for Join-count.
ncod	Number of elements coding each category (e.g., if x ncod =1, if xx, ncod = 2, and so on). Only for Join-count analysis.
ploidy	Ploidy of genetic data. Only for for Join-count analysis.
nsim	Number of Monte-Carlo simulations.
alternative	The alternative hypothesis. If "auto" is selected (default) the program determines the alternative hypothesis. Other options are: "two.sided", "greater" and "less".
adjust	Correction method of P-values for multiple tests, passed to p.adjust. Defalut is "fdr".
row.sd	Logical. should be row standardized the matrix? Default FALSE (binary weights).
plotit	Should be printed a histogram of the simulation? Defalut TRUE.

Value

The program returns an object of class "eco.gsa" with the following slots:

> METHOD method used in the analysis

> OBS observed value when a single variable is tested

> EXP expected value when a single variable is tested

> PVAL P-value when a single variable is tested

> ALTER alternative hypotesis when a single variable is tested

> NSIM number of simulations

> MULTI table with observed and expected values, P-values and alternative hypoteses when multiple variables are tested

ACCESS TO THE SLOTS The content of the slots can be accessed with the corresponding accessors, using the generic notation of EcoGenetics (<ecoslot.> + <name of the slot> + <name of the object>). See help("EcoGenetics accessors") and the Examples section below.

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

References

Geary R. 1954. The contiguity ratio and statistical mapping. The incorporated statistician, 115-146.

Moran P. 1950. Notes on continuous stochastic phenomena. Biometrika, 17-23.

Reich R., R. Czaplewski and W. Bechtold. 1994. Spatial cross-correlation of undisturbed, natural shortleaf pine stands in northern Georgia. Environmental and Ecological Statistics, 1: 201-217.

Sokal R. and N. Oden 1978. Spatial autocorrelation in biology: 1. Methodology. Biological journal of the Linnean Society, 10: 199-228.

Sokal R. and N. Oden. 1978. Spatial autocorrelation in biology. 2. Some biological implications and four applications of evolutionary and ecological interest. Biological Journal of the Linnean Society, 10: 229-49.

Examples

```
## Not run:

data(eco.test)

# Moran's I

### one test
con <- eco.weight(eco[["XY"]], method = "circle", d1 = 0, d2 = 2)
global <- eco.gsa(Z = eco[["P"]][, 1], con = con, , method = "I", nsim = 200)
global

require(adegenet)
con2<-chooseCN(eco[["XY"]], type = 1, result.type = "listw", plot.nb = FALSE)
global <- eco.gsa(Z = eco[["P"]][, 1], con = con2, , method = "I", nsim = 200)
global

#-----------------------
# ACCESSORS USE EXAMPLE
#-----------------------
```

```
# the slots are accesed with the generic format
# (ecoslot. + name of the slot + name of the object).
# See help("EcoGenetics accessors")

# observed value
ecoslot.OBS(global)

# p-value
ecoslot.PVAL(global)

#----------------
# multiple tests
#----------------

con <- eco.weight(eco[["XY"]], method = "circle", d1 = 0, d2 = 2)
global <- eco.gsa(Z = eco[["P"]], con = con, , method = "I", nsim = 200)
global

#--------------------------------
# accessor use in multiple tests
#--------------------------------

ecoslot.MULTI(global)


#----------------------------------------

# Gearys's C

con <- eco.weight(eco[["XY"]], method = "circle", d1 = 0, d2 = 2)
global.C <- eco.gsa(Z = eco[["P"]][, 1], con = con, method = "C", nsim = 200)
global.C

#----------------------------------------

# Bivariate's Moran's Ixy

con <- eco.weight(eco[["XY"]], method = "circle", d1 = 0, d2 = 2)
global.Ixy <- eco.gsa(Z = eco[["P"]][, 1], Y = eco[["E"]][, 1],
con = con, method = "CC", nsim = 200)
global.Ixy

#----------------------------------------

# Join-count

## using the allelic frequency matrix of an ecogen object.
## The data is diploid. Frequencies are transformed into counts
## as ploidy * frequency_matrix:

Z = 2* eco[["A"]]
```

```
con <- eco.weight(eco[["XY"]], method = "circle", d1 = 0, d2 = 2)

# using the first allele of the matrix
global.JC <- eco.gsa(Z[, 1], ncod = 1, con = con, method = "JC", nsim = 5)
global.JC
# Note that with large data sets, join-count estimation can be slow.

# counting joins between genotypes of the locus 1:
global.JC <- eco.gsa(Z = eco[["G"]][,1], ploidy = 2, con = con, method = "JC", nsim = 1)
global.JC

## End(Not run)
```

`eco.kin.loiselle` *Obtention of the multilocus Loiselle's Fij matrix*

Description

Obtention of the multilocus Loiselle's Fij matrix

Usage

```
eco.kin.loiselle(eco)
```

Arguments

`eco`	Object of class ecogen.

References

Kalisz, S., J. Nason, F.M. Handazawa, and S. Tonsor. 2001. Spatial population genetic structure in Trillium grandiflorum: the roles of dispersal, mating, history, and selection. Evolution 55: 1560-1568.

Loiselle, B., V. Sork, J. Nason, and C. Graham. 1995. Spatial genetic structure of a tropical understory shrub, Psychotria officinalis (Rubiaceae). American Journal of Botany 1420-1425.

Examples

```
## Not run:

data(eco.test)
loiselle <- eco.kin.loiselle(eco)
loiselle[1:5, 1:5]


## End(Not run)
```

`eco.lagweight` *Obtention of a list of spatial weights for classes defined by inter-individual distances or nearest-neighbors*

Description

This program returns a list of weights matrices (binary or row-standardized), one for each spatial class. For a given maximum and minimum inter-individual distance (IID), the data can be partitioned in different ways. The program set as default the highest IID as the maximum, and the lowest as the minimum. These values can be changed with "smax" and "smin", respectively. Intervals may be generated with the parameters "int" (which divides the range each int distance units), "nclass" (which divides the range in n-classes) and "size" (a fixed size of pairs included in each class). When a partition argument is not given (int, nclass or size) the program determines the number of classes using the Sturge's rule (default) or the Freedman- Diaconis method. Two additional methods can be used: a list with nearest-neighbors matrices, from 1 to k nearest-neighbors, may be generated with the argument "kmax". A custom vector with breaks may be provided by the user with the argument "seqvec". See the examples.

Usage

```
eco.lagweight(XY, int = NULL, smin = 0, smax = NULL, kmax = NULL,
  nclass = NULL, seqvec = NULL, size = NULL, bin = c("sturges", "FD"),
  cummulative = FALSE, row.sd = FALSE, self = FALSE, latlon = FALSE)
```

Arguments

`XY`	Matrix/data frame with projected coordinates.
`int`	Distance interval in the units of XY.
`smin`	Minimum class distance in the units of XY.
`smax`	Maximum class distance in the units of XY.
`kmax`	Number of nearest-neighbors.
`nclass`	Number of classes.
`seqvec`	Vector with breaks in the units of XY.
`size`	Number of individuals per class.
`bin`	Rule for constructing intervals when a partition parameter (int, nclass or size) is not given. Default is Sturge's rule (Sturges, 1926). Other option is Freedman-Diaconis method (Freedman and Diaconis, 1981).
`cummulative`	Logical. Should be created matrices considering cummulative distances instead of discrete classes? Default FALSE.
`row.sd`	Logical. Should be row standardized each matrix? Default FALSE (binary weights).
`self`	Logical. Should be included a first matrix with ones in the diagonal and zeros zeros in the other cells? Default FALSE.

`latlon` Are the coordinates in decimal degrees format? Defalut FALSE. If TRUE, the coordinates must be in a matrix/data frame with the longitude in the first column and latitude in the second. The position is projected onto a plane in meters with the function `geoXY`.

Value

The program returns an object of class "eco.lagweight" with the following slots:

> PAR parameters used for the construction of breaks

> PAR.VAL values of the parameters used for the construction of breaks

> ROW.SD row standardization (logical)

> SELF data self-included (logical)

> W weights list

> XY coordinates

> MEAN mean class distances

> LOGMEAN mean of the class distances logarithm

> CARDINAL number of elements in each class

> BREAKS breaks

> METHOD breaks construction method

ACCESS TO THE SLOTS The content of the slots can be accessed with the corresponding accessors, using the generic notation of EcoGenetics (<ecoslot.> + <name of the slot> + <name of the object>). See help("EcoGenetics accessors") and the Examples section below.

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

References

Freedman D., and P. Diaconis. 1981. On the histogram as a density estimator: L 2 theory. Probability theory and related fields, 57: 453-476.

Sturges H. 1926. The choice of a class interval. Journal of the American Statistical Association, 21: 65-66.

Examples

```
## Not run:
data(eco.test)

# method sturges-smax: in this case, the program generates
# classes using the Sturge's rule.
# As smax and smin are undefined, the program uses the default
# options (smin = 0, and smax = maximum inter-individual distance)
classlist <- eco.lagweight(eco[["XY"]])
classlist
```

```
# method sturges-smax: idem, but smax = 16
classlist <- eco.lagweight(eco[["XY"]], smax=16)

## using smax <16 in this case generates empty classes,
## which is not allowed

# method sturges-smax: idem, but smin = 3
classlist <- eco.lagweight(eco[["XY"]], smin = 3, smax = 15)

#-----------------------
# ACCESSORS USE EXAMPLE
#-----------------------

# the slots are accessed with the generic format
# (ecoslot. + name of the slot + name of the object).
# See help("EcoGenetics accessors")

ecoslot.BREAKS(classlist) # information about breaks. It includes the upper and lower limits

# method sturges-smax: complete range,
# and cummulative = TRUE (instead of using
# lower and upper limits for each class, only the upper is used in turn)
classlist <- eco.lagweight(eco[["XY"]], cummulative = TRUE)

# method n.classes-smax: complete range partitioned in 4 classes
classlist <- eco.lagweight(eco[["XY"]], nclass = 4)

# method n.classes-smax: idem, but smax =  15
classlist <- eco.lagweight(eco[["XY"]], nclass = 4, smax = 15)

# method int-smax: the complete range partitioned each <int> units
# of inter-individual distance
classlist <- eco.lagweight(eco[["XY"]], int = 2)

# method int-smax: idem, but smax = 15 and smin = 3
classlist <- eco.lagweight(eco[["XY"]], int = 2, smin = 3, smax = 15)

# method equal.size: n individuals in each class,
# partitioning the complete range.
classlist <- eco.lagweight(eco[["XY"]], size = 1000)

## In the latter example, as an inter-individual distance
## appear more than one time (different individuals pairs,
## identical distances), with a size <700 the limits
## of some classes cannot be defined, and this is not allowed

# method equal.size: n individuals in each class,
# but smax = 15
classlist <- eco.lagweight(eco[["XY"]], size = 1000, smax = 15)

# method kmax: sequence from k = 1 to k = n, in this case, n = 3
classlist <- eco.lagweight(eco[["XY"]], kmax = 3)
```

```
# method kmax: idem, but elements self-included
# (i.e., the pairs i-i, for all individuals i, are included)
classlist <- eco.lagweight(eco[["XY"]], kmax = 3, self = TRUE)

# method seqvec: a vector with the breaks is used
vec <- seq(0, 10, 2)
classlist <- eco.lagweight(eco[["XY"]], seqvec = vec)

## End(Not run)
```

`eco.lmtree` *Fitting Multiple Linear Regression models by stepwise AIC selection and Multiple Classification and Regression Trees via party*

Description

This program fits for each dependent variable, a Multiple Linear Regression model calling the function `step` for choosing the best model by AIC criterion, or a Multiple Classification and Regression Trees model, using the package party. The summary of the model returns information about the significance of the models, F-statistics and degrees of freedom, when is fitted a "mlm"; otherwise, when the model fitted is a "mctree", the summary returns the plots of those trees with significant splits.

Usage

```
eco.lmtree(df1, df2, analysis = c("mlm", "mctree"), mod.class = "+",
  fact = NULL, ...)
```

Arguments

`df1`	Data frame with dependent variables as columns.
`df2`	Data frame with independent variables as columns.
`analysis`	Class of analysis to perform. "mlm" for multiple linear regression analysis, or "mctree" for a multiple classification tree analysis.
`mod.class`	"+" for additive model, "*" for model with interaction, in both cases, these models will include all terms in the dependent data frame. If other model than these two is desired, it could be specified as a string with the names of those columns of the independent variable that should be used as terms. This string corresponds to the right side "x" of a formula y ~ x (see examples).
`fact`	Optional factor for estimating the frequencies of individuals from different levels in each node, when the analysis performed is "mctree".
`...`	Further arguments passed to `lm` or `ctree`

Value

When the analysis selected is "mlm", the output object has three main slots:

> MLM: the results of the model

> SUMMARY.MLM the summary for each variable returned by the lm function

> ANOVA.MLM with the ANOVAs results.

When the analysis selected is "mctree", the output object has also three main slots:

> TREES: Trees returned by the multiple ctree analysis.

> PREDICTIONS: Predictions of the analysis.

> FREQUENCIES: Number of individuals predicted in each node.

ACCESS TO THE SLOTS The content of the slots can be accessed with the corresponding accessors, using the generic notation of EcoGenetics (<ecoslot.> + <name of the slot> + <name of the object>). See help("EcoGenetics accessors") and the Examples section below

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

References

Hothorn T., K. Hornik, and A. Zeileis. 2006. Unbiased Recursive Partitioning: A Conditional Inference Framework. Journal of Computational and Graphical Statistics, 15: 651-674.

Examples

```
## Not run:

data(eco2)

# mlm additive model
mod <- eco.lmtree(df1 = eco[["P"]], df2 = eco[["E"]], analysis = "mlm")
mod
summary(mod)

# mctree additive model
mod <- eco.lmtree(df1 = eco[["P"]], df2 = eco[["E"]],
analysis = "mctree", fact = eco[["S"]]$pop)


#-----------------------
# ACCESSORS USE EXAMPLE
#-----------------------

# the slots are accessed with the generic format
# (ecoslot. + name of the slot + name of the object).
# See help("EcoGenetics accessors")

summary(mod)
```

```
ecoslot.FREQUENCIES(mod)        # slot FREQUENCIES

# frequency table with counts of individuals in populations x terminal nodes
tabfreq <- do.call(cbind, ecoslot.FREQUENCIES(mod))
namestab <- lapply(ecoslot.FREQUENCIES(mod), ncol)
namestab <- lapply(namestab, rep)
namestab <- rep(names(namestab), namestab)
colnames(tabfreq) <- namestab
tabfreq

# mlm custom model
mymod <- "E1+E2*E3"
mod <- eco.lmtree(df1 = eco[["P"]], df2 = eco[["E"]], analysis = "mlm", mod.class = mymod)
summary(mod)

# mctree custom model
mod <- eco.lmtree(df1 = eco[["P"]], df2 = eco[["E"]],
analysis = "mctree", mod.class = mymod, fact = eco[["S"]]$pop)


summary(mod)


## End(Not run)
```

`eco.lsa` *Local spatial analysis*

Description

This program computes Getis-Ord G and G*, and LISA's (local Moran and local Geary) statistics for the data Z, with P-values or bootstrap confidence intervals.

Usage

```
eco.lsa(Z, con, method = c("G*", "G", "I", "C"), zerocon = NA, nsim = 99,
  conditional = c("auto", TRUE, FALSE), test = c("permutation",
  "bootstrap"), alternative = c("auto", "two.sided", "greater", "less"),
  adjust = "none")
```

Arguments

`Z`	Vector for the analysis.
`con`	An object of class eco.weight obtained with the function `eco.weight`, a "listw" object, or a matrix, containing the weights for the analysis. If a matrix, an attribute "xy" with the projected coordinates is required.
`method`	Method of analysis: "G" for Getis-Ord G, "G*" for Getis-Ord G*, "I" for local Moran's I or "C" for local Geary's C.

zerocon	If zerocon = 0 the program assigns the value 0 to those individuals with no connections; if zerocon = NA the program assigns NA. Default is NA.
nsim	Number of Monte-Carlo simulations.
conditional	Logical. Should be used a conditional randomization? (Anselin 1998, Sokal and Thomson 2006). The option "auto" sets conditional = TRUE for LISA methods and G, as suggested by Sokal (2008).
test	If test = "bootstrap", for each individual test, the program generates a bootstrap resampling and the associated confidence intervals of the null hypotesis. If test = "permutation" (default) a permutation test is made and the P-value is computed.
alternative	The alternative hypothesis for "permutation" test. If "auto" is selected (default) the program determines the alternative hypothesis in each individual test. Other options are: "two.sided", "greater" and "less".
adjust	Correction method of P-values for multiple tests, passed to p.adjust. Defalut is "none" (no correction).

Value

The program returns an object of class "eco.lsa" with the following slots:

> OUT results

> METHOD method (coefficent) used in the analysis

> TEST test method used (bootstrap, permutation)

> NSIM number of simulations

> PADJUST P-values adjust method for permutation tests

> COND conditional randomization (logical)

> XY input coordinates

ACCESS TO THE SLOTS The content of the slots can be accessed with the corresponding accessors, using the generic notation of EcoGenetics (<ecoslot.> + <name of the slot> + <name of the object>). See help("EcoGenetics accessors") and the Examples section below

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

References

Anselin L. 1995. Local indicators of spatial association-LISA. Geographical analysis. 27: 93-115.

Getis A., and J. Ord. 1992. The analysis of spatial association by use of distance statistics. Geographical analysis, 24: 189-206.

Ord J., and A. Getis. 1995. Local spatial autocorrelation statistics: distributional issues and an application. Geographical analysis, 27: 286-306.

Sokal R., N. Oden and B. Thomson. 1998. Local spatial autocorrelation in a biological model. Geographical Analysis, 30: 331-354.

Sokal R. and B. Thomson. 2006. Population structure inferred by local spatial autocorrelation: an example from an Amerindian tribal population. American journal of physical anthropology, 129: 121-131.

Examples

```
## Not run:

data(eco.test)

#####################
# GETIS-ORD'S G*
#####################

con<- eco.weight(eco[["XY"]], method = "knearest",  k = 4, self = TRUE) # self = TRUE for G*
getis.ak <- eco.lsa(eco[["P"]][, 1], con, method = "G*", nsim = 99, adjust = "none")
getis.ak

### to plot the results, the function "eco.lsa" calls "eco.rankplot"
(see ?eco.rankplot) when test = "permutation" and "eco.forestplot" (see ?eco.forestplot)
 when test = "bootstrap"
p <- plot(getis.ak)      # rankplot graph
p    #  points with colors of the color-scale:
     #  points with P < 0.05. Yellow points : points with P > 0.05
p <- plot(getis.ak, significant = FALSE)
p    # all points have a color of the color-scale

#-----------------------
# ACCESSORS USE EXAMPLE
#-----------------------

# the slots are accessed with the generic format
# (ecoslot. + name of the slot + name of the object).
# See help("EcoGenetics accessors")

ecoslot.OUT(getis.ak)

## bootstrap example
getis.akb <- eco.lsa(eco[["P"]][, 1], con, method = "G*", nsim = 99, test = "bootstrap")
p <- plot(getis.akb)      # forestplot graph
p + ggplot2::theme_bw()   # the plot can be modified with ggplot2
                          # In this case, the background is modified  (white color)

#-------------------------------------------------------------------------------#

#####################
# GETIS-ORD'S G
#####################

con <- eco.weight(eco[["XY"]], method = "knearest", k = 4)
# self = FALSE for G
getis <- eco.lsa(eco[["P"]][, 1], con, method = "G", nsim = 99)
plot(getis)

#-------------------------------------------------------------------------------#

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

```
# LOCAL MORAN'S I
#####################

#-------------------------
# TESTING PHENOTYPIC DATA-
#-------------------------

con <- eco.weight(eco[["XY"]], method = "knearest",  k = 4, row.sd = TRUE)
# row standardized weights = TRUE

# test for the first trait of the data frame P
localmoran <- eco.lsa(eco[["P"]][, 1], con, method = "I", nsim = 99)

plot(localmoran)

# test for several variables

all.traits <- apply(eco[["P"]], 2, eco.lsa,  con, method = "I", nsim = 99)

# Observed statistic and P-values tables (individuals x traits)
stat.P <- sapply(all.traits, function(x) return(ecoslot.OUT(x)[,1]))
pval.P <- sapply(all.traits, function(x) return(ecoslot.OUT(x)[,4]))

# Plot of the phenotypic spatial patterns

par(mfrow = c(2,4))
for(i in 1:8) {
image(matrix(stat.P[,i], 15,15))
}

par(mfrow = c(2,4))
for(i in 1:8) {
image(matrix(pval.P[,i], 15,15))
}


#-------------------------
# TESTING GENOTYPIC DATA-
#-------------------------

# eco[["A"]] is a matrix with the genetic data of "eco"
# as frequencies for each allele in each individual.

head(eco[["A"]])      # head of the matrix - 40 alleles

con <- eco.weight(eco[["XY"]], method = "knearest",  k = 4, row.sd = TRUE)
# row standardized weights = TRUE

# test for a single allele
localmoran.geno <-  eco.lsa(eco[["A"]][, 32], con, method = "I", nsim = 99)

# test for several alleles -  40 alleles (it runs in less than 1 min
# for 99 simulations per allele;  999 simulations takes ~ 11 s per allele,
```

```
# less than 8 min in total.)
all.alleles <- apply(eco[["A"]], 2, eco.lsa,  con, method = "I", nsim = 99)

# plot all alleles to get an overview of the spatial patterns
lapply(all.alleles, plot)

# Observed statistic and P-values tables (individuals x loci)
stat.G <- sapply(all.alleles, function(x) return(ecoslot.OUT(x)[,1]))
pval.G <- sapply(all.alleles, function(x) return(ecoslot.OUT(x)[,4]))

# counting individuals with P < 0.05 for each allele (5 * 225 /100 ~  12 significant tests
# by random)
signif <- lapply(all.alleles, function(x) sum(ecoslot.OUT(x)[,4] < 0.05))
signif <- unlist(signif)

# filtering alleles, loci with > 12 significant individual tests

A.local <- eco[["A"]][, signif > 12]     #filtered matrix
stat.G.f <- stat.G[, signif > 12]
pval.G.f <- pval.G[, signif > 12]

# Plot of the genotypic spatial patterns

# one plot possibility, using the EcoGenetics method <rankplot>
all.local <- all.alleles[signif > 12]
lapply(all.local, plot)

# other plot possibility, using <image>
par(mfrow = c(3,4))
for(i in 1:12) {
image(matrix(stat.G[,i], 15,15))
}

par(mfrow = c(3,4))
for(i in 1:12) {
image(matrix(pval.G[,i], 15,15))
}


#-------------------------------------------------------------------------#

#####################
# LOCAL GEARY'S C
#####################

con<- eco.weight(eco[["XY"]], method = "knearest",  k = 4, row.sd = TRUE)
# row standardized weights = TRUE
localgeary <- eco.lsa(eco[["P"]][, 1], con, method = "C", nsim = 99, adjust = "none")
plot(localgeary)


## End(Not run)
```

`eco.malecot` *Global and local kinship analyses (beta version)*

Description

NOTE: THIS IS A BETA VERSION OF THE FUNCTION (UNDER DEVELOPMENT). The program computes a global multilocus correlogram, or a local analysis, using a kinship matrix. When a kinship matrix is not given as input, the program computes the Loiselle's Fij (Kalisz et al., 2001; Loiselle et al., 1995).

Usage

```
eco.malecot(eco, method = c("global", "local"), kinmatrix = NULL,
  int = NULL, smin = 0, smax = NULL, nclass = NULL, kmax = NULL,
  seqvec = NULL, size = NULL, type = c("knearest", "radialdist"),
  cubic = TRUE, testclass.b = TRUE, testmantel.b = TRUE,
  jackknife = TRUE, cummulative = FALSE, nsim = 99,
  test = c("permutation", "bootstrap"), alternative = c("auto", "two.sided",
  "greater", "less"), sequential = TRUE, conditional = c("AUTO", "TRUE",
  "FALSE"), bin = c("sturges", "FD"), row.sd = FALSE, adjust = "holm",
  latlon = FALSE)
```

Arguments

`eco`	Object of class ecogen.
`method`	Analysis method: "global" or "local".
`kinmatrix`	Alternative kinship matrix. The program computes the Loiselle's kinship matrix (codominant data) with the genetic information of the ecogen object if kinmatrix = NULL (Defaul option).
`int`	Distance interval in the units of XY.
`smin`	Minimum class distance in the units of XY.
`smax`	Maximum class distance in the units of XY.
`nclass`	Number of classes.
`kmax`	Number of nearest-neighbors for local analysis.
`seqvec`	Vector with breaks in the units of XY.
`size`	Number of individuals per class.
`type`	Weight mode for local analyis: "knearest" for nearest neigbors, "radialdist" for radial distances. Default is knearest.
`cubic`	Should be performed a cubic interpolation (res~ ln(dij)) with the regression residuals (res) of (kinship)ij ~ ln(dij) ? Defalut TRUE.
`testclass.b`	Should be performed a permutation test in each individual class? Defalut TRUE.
`testmantel.b`	Should be performed a Mantel test for testing the slope (b)? Defalut TRUE.

jackknife	Should be performed jackknife in each individual class for computing the standard deviation (SD) of the coancestry (class) values? Defalut TRUE.
cummulative	Should be construced a cummulative correlogram?.
nsim	Number of Monte-Carlo simulations.
test	If test = "bootstrap", the program generates a bootstrap resampling and the associated confidence intervals of the null hypothesis. If test = "permutation" (default) a permutation test is made and the P-values are computed.
alternative	The alternative hypothesis. If "auto" is selected (default) the program determines the alternative hypothesis. Other options are: "two.sided", "greater" and "less".
sequential	Should be performed a Holm-Bonberroni (Legendre and Legendre, 2012) adjustment of P-values for global analysis? Defalult TRUE.
conditional	Logical. Should be used a conditional randomization? (Anselin 1998, Sokal and Thomson 2006). The option "auto" sets conditional = TRUE for LISA methods and G, as suggested by Sokal (2008).
bin	Rule for constructing intervals when a partition parameter (int, nclass or size) is not given. Default is Sturge's rule (Sturges, 1926). Other option is Freedman-Diaconis method (Freedman and Diaconis, 1981).
row.sd	Logical. Should be row standardized the matrix? Default FALSE (binary weights).
adjust	P-values correction method for multiple tests passed to p.adjust. Defalut is "holm".
latlon	Are the coordinates in decimal degrees format? Defalut FALSE. If TRUE, the coordinates must be in a matrix/data frame with the longitude in the first column and latitude in the second. The position is projected onto a plane in meters with the function geoXY.

Details

The GLOBAL ANALYSIS mode, computes a multilocus correlogram, with a detailed summary (see the content of the slot OUT in the "return" section). It also computes (see details about the slot SP in the "return" section): - the slope of the kinship individual values vs the logarithm of the distance, (kinship)ij ~ ln(dij), with a jackknife confidence interval - a Mantel test for testing the association between (kinship)ij and ln(dij) - The Sp statistic (Vekemans and Hardy, 2004) with confidence intervals - A cubic interpolation of (kinship)ij ~ ln(dij) residuals vs ln(dij)

The LOCAL ANALYSIS mode, computes a local kinship estimate, based in a weighted mean (for the each individual). The signification of each local statistic is computed using a permutation test, as in eco.lsa (see ?"eco.lsa"). Default option do not adjust the individual P values for multiple corrections.

Value

NOTE: THIS IS A BETA VERSION OF THE FUNCTION (UNDER DEVELOPMENT).

For the global analysis, the program returns an object of class "eco.IBD" with the following slots:

> OUT analysis output.

In the permutation test case contains: - d.mean: mean class distance; - d.log: mean logarithm of the class distance; - obs, exp, alter, p.val: observed, and expected value of the statistic under

randomization, alternative, P value; - mean.jack, sd.jack, Jack.CI.inf, Jack.CI.sup: jackknifed mean and SD, and confidence intervals for the statistic; - null.lwr, nul.uppr: lower and upper bound of the jackknife confidence interval for the statistic; - cardinal: number of individuals in each class;

In the bootstrap test case contains: - d.mean: mean class distance; - d.log: mean logarithm of the class distance; - obs: observed value of the statistic; - mean.jack, sd.jack, Jack.CI.inf, Jack.CI.sup: jackknifed mean and SD, and confidence intervals for the statistic; - null.lwr, nul.uppr: lower and upper bound of the jackknife confidence interval for the statistic; - cardinal: number of individuals in each class;

> IN analysis input data

> SP Sp statistic results

It contains:

- the regression model; - information about the distance interval used for the regression (restricted); - slope (bhat) information (bhat = estimate, SD= bhat jacknife SD, theta = bhat jackknife mean, CI 5% and 95% = 95% confidence interval for bhat); - X-intercept = dij intercept (in the original units) for the line with slope "bhat", F1 = first class statistic value, and F1 5% and 95% = confidence interval for the first class statistic; - mantel.obs.b = observed value of the Mantel test between kinship(Fij) and ln(dij); mantel.pval.b = Mantel test P value; - sp = Sp statistics (sp = Sp observed value, CI 5% and 95% = 95% confidence interval for Sp); - cubic_model = cubic model for (kinship)ij ~ ln(dij) r esiduals vs ln(dij);

> BEAKS breaks

> CARDINAL number of elements in each class

> NAMES variables names

> METHOD analysis method

> DISTMETHOD method used in the construction of breaks

> TEST test method used (bootstrap, permutation)

> NSIM number of simulations

> PADJUST P-values adjust method for permutation tests

——

For the local analysis, the program returns an object of class "eco.lsa" with the following slots:

> OUT results

> In the permutation test case contains:

- d.mean: mean class distance - obs, exp, alter, p.val: observed, and expected value of the statistic under randomization, alternative, P value; - null.lwr, nul.uppr: lower and upper bound of the jackknife confidence interval for the statistic; - cardinal: number of individuals in each class;

> In the bootstrap test case contains: - d.mean: mean class distance; - obs: observed value of the statistic; - null.lwr, nul.uppr: lower and upper bound of the jackknife; confidence interval for the statistic; - cardinal: number of individuals in each class;

> METHOD method (coefficent) used in the analysis

> TEST test method used (bootstrap, permutation)

> NSIM number of simulations

> PADJUST P-values adjust method for permutation tests

> COND conditional randomization (logical)

> XY input coordinates

ACCESS TO THE SLOTS The content of the slots can be accessed with the corresponding accessors, using the generic notation of EcoGenetics (<ecoslot.> + <name of the slot> + <name of the object>). See help("EcoGenetics accessors") and the Examples section below.

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

References

Double M., R. Peakall, N. Beck, and Y. Cockburn. 2005. Dispersal, philopatry, and infidelity: dissecting local genetic structure in superb fairy-wrens (Malurs cyaneus). Evolution 59: 625-635.

Kalisz S., J. Nason, F.M. Handazawa, and S. Tonsor. 2001. Spatial population genetic structure in Trillium grandiflorum: the roles of dispersal, mating, history, and selection. Evolution 55: 1560-1568.

Loiselle B., V. Sork, J. Nason, and C. Graham. 1995. Spatial genetic structure of a tropical understory shrub, Psychotria officinalis (Rubiaceae). American Journal of Botany 1420-1425.

Vekemans, X., and O. Hardy. 2004. New insights from fine-scale spatial genetic structure analyses in plant populations. Molecular Ecology, 13: 921-935.

Examples

```
## Not run:

data(eco.test)

# ---global analysis---

globaltest <- eco.malecot(eco=eco, method = "global", smax=10,
                          size=1000)
plot(globaltest)    # Significant mean class coancestry classes at
                    # individual level (alpha = 0.05,
                    # out of the red area),
                    # and family-wise P corrected values (red-blue
                    # points, indicated in the legend)

# ecoslot.SP(globaltest) contains:
# - the slope (bhat) and values with confidence intervals
#  of the regression reg = kinship ~ ln(distance_between_individuals)
#- A Mantel test result for assesing the relation between
#  between  kinship and ln(distance_between_individuals)
#- A cubic interpolation between the residuals of reg and
#  ln(distance_between_individuals)
#- the sp statistic and its confidence interval

# ecoslot.OUT(globaltest) contains:
# - In permutation case, the values of mean and log-mean distance
#   classes; observed class value; expected + alternative + P value,
```

```
#   the bootstrap null confidence intervals and
#   jackknife statistics (jackknifed mean, jackknifed SD, and
#                         CI for the class statistic)

# - In bootstrap case, the values of mean and log-mean distance
#   classes;the bootstrap null confidence intervals and
#   jackknife statistics (jackknifed mean, jackknifed SD, and
#                         CI for the class statistic)

#-----------------------------------------------------------#
# ---local analysis---


(using the spatial weights).

# ---local analysis with k nearest neighbors---



localktest <- eco.malecot(eco=eco, method = "local",
                          type = "knearest", kmax = 5,
                          adjust = "none")
plot(localktest)


# ---local analysis with radial distance---

localdtest <- eco.malecot(eco=eco, method = "local",
                         type = "radialdist", smax = 3,
                         adjust = "none")

plot(localdtest)                         # rankplot graphic (see ?"eco.rankplot")

                                         # Significant values
                                         # in blue-red scale,
                                         # non significant
                                         # values in yellow

plot(localktest, significant = FALSE)    # significant and non
                                         # signficant values
                                         # in blue-red scale

# The slot OUT of localktest (ecoslot.OUT(localktest)) and localdtest
# (ecoslot.OUT(localdtest)) contains:
# - the mean distance per individual, observed value of the
#   statistic, expected + alternative + P value + null hypotesis
#   confidence intervals,  or boostrap confidence intervals in
#   permutation or bootstrap cases, respectively.

## End(Not run)
```

`eco.mantel` *Mantel and partial Mantel tests*

Description

This program computes the Mantel test between the distance matrices d1 and d2, or a partial Mantel test between the distance matrices d1 and d2, conditioned on dc.

Usage

```
eco.mantel(d1, d2, dc = NULL, method = c("pearson", "spearman", "kendall"),
  nsim = 99, alternative = c("auto", "two.sided", "less", "greater"), ...)
```

Arguments

`d1`	Distance matrix.
`d2`	Distance matrix.
`dc`	Distance matrix (optional).
`method`	Correlation method used for the construction of the statistic ("pearson", "spearman" or "kendall"). Kendall's tau computation is slow.
`nsim`	Number of Monte-Carlo simulations.
`alternative`	The alternative hypothesis. If "auto" is selected (default) the program determines the alternative hypothesis. Other options are: "two.sided", "greater" and "less".
`...`	Additional arguments passed to `cor`.

Value

An object of class "eco.gsa" with the following slots:

> METHOD method used in the analysis

> OBS observed value

> EXP expect value

> PVAL P-value

> ALTER alternative hypotesis

> NSIM number of simulations

ACCESS TO THE SLOTS The content of the slots can be accessed with the corresponding accessors, using the generic notation of EcoGenetics (<ecoslot.> + <name of the slot> + <name of the object>). See help("EcoGenetics accessors") and the Examples section below

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

References

Legendre P. 2000. Comparison of permutation methods for the partial correlation and partial Mantel tests. Journal of Statistical Computation and Simulation, 67: 37-73.

Legendre P., and M. Fortin. 2010. Comparison of the Mantel test and alternative approaches for detecting complex multivariate relationships in the spatial analysis of genetic data. Molecular Ecology Resources, 10: 831-844.

Mantel N. 1967. The detection of disease clustering and a generalized regression approach. Cancer research, 27: 209-220.

Smouse P. J. Long and R. Sokal. 1986. Multiple regression and correlation extensions of the Mantel test of matrix correspondence. Systematic zoology, 627-632.

Examples

```
## Not run:

data(eco.test)

eco.mantel(d1 = dist(eco[["P"]]), d2 = dist(eco[["E"]]), nsim = 99)  # ordinary Mantel test

pm <- eco.mantel(d1 = dist(eco[["P"]]), d2 = dist(eco[["E"]]),
dc = dist(eco[["XY"]]), nsim = 99)                                  # partial Mantel test

#-----------------------
# ACCESSORS USE EXAMPLE
#-----------------------

# the slots are accessed with the generic format
# (ecoslot. + name of the slot + name of the object).
# See help("EcoGenetics accessors")

ecoslot.OBS(pm)     # slot OBS (observed value)
ecoslot.PVAL(pm)    # slot PVAL (P-value)


## End(Not run)
```

`eco.merge` *Merging two ecogen objects. Ordering the rows of an ecogen object according to the rows of another*

Description

Merging two ecogen objects. Ordering the rows of an ecogen object according to the rows of another

Usage

```
eco.merge(e1, e2, ...)
```

Arguments

e1	Object of class "ecogen".
e2	Object of class "ecogen".
...	Data frames to merge. Could be any combination of the following: "XY",P","G","E" and "C", or "ALL". If a "G" data frame is provided, the program generates also the INT slot coding the missing data as "0".

Details

This program generates an ecogen object binding the columns of the individuals with matching row names in e1 and e2. If the objects have different number of rows, the result is a merged data frame with the rows in the order of the first object. If the objects have the same number of rows, but in a different order, the product is an object with the rows ordered as the first object. The algorithm matches sequentially the data frame pairs of each slot that the user wishes to merge.

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

Examples

```
## Not run:
data(eco.test)
eco
eco1 <- eco
eco1[["XY"]] <- eco[["XY"]][sample(1:225), ]  #object with permuted rows
eco[["XY"]]
eco1[["XY"]]
merged <- eco.merge(eco, eco1)
merged

## End(Not run)
```

---

eco.NDVI *Generating atmospherically corrected NDVI and MSAVI2 images for temporal series of Landsat 5 and 7*

Description

This program generates atmospherically corrected images of NDVI and MSAVI2. The images of multiple dates can be processed in a single run. The bands 4 and 3 of each date (previously subsetted to the region of analysis) must be in the working directory. A table with information for each image as described in the parameter tab (see also the example) is needed for processing the information.

Usage

```
eco.NDVI(tab, correct = c("COST", "DOS"), method = c("NDVI", "MSAVI2"),
  landsat = c("LT5", "LT7.L", "LT7.H"), datatype = c("FLT4S", "FLT8S",
  "INT4U", "INT4S", "INT2U", "INT2S", "INT1U", "INT1S", "LOG1S"))
```

Arguments

tab	data.frame with 7 columns: The date of the images (format: YYYY/MM/DD), the sun elevation (both values could be extracted from Landsat headers), the name of the band 4, the name of the band 3, the starting haze value of the band 4, the starting haze value of the band 3, and the name of the output file. Each row corresponds to an image of different date.
correct	Correction method ("COST", "DOS").
method	Vegetation index ("NDVI", "MSAVI2").
landsat	Satellite data source ("LT5" for Landsat 5, "LT7.L" for Landsat 7 low gain and "LT7.H" for Landsat 7 high gain).
datatype	type of data, see dataType. Default "FLT4S".

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

References

Chander G., B. Markham, and D. Helder. 2009. Summary of current radiometric calibration coefficients for Landsat MSS, TM, ETM+, and EO-1 ALI sensors. Remote sensing of environment, 113: 893-903.

Chavez P. 1989. Radiometric calibration of Landsat Thematic Mapper multispectral images. Photogrammetric Engineering and Remote Sensing, 55: 1285-1294.

Chavez P. 1996. Image-based atmospheric corrections-revisited and improved. Photogrammetric engineering and remote sensing, 62: 1025-1035.

Goslee S. 2011. Analyzing remote sensing data in R: the landsat package. Journal of Statistical Software, 43: 1-25.

Song C., C. Woodcock, K. Seto, M. Lenney and S. Macomber. 2001. Classification and change detection using Landsat TM data: when and how to correct atmospheric effects?. Remote sensing of Environment, 75: 230-244.

Tucker C. 1979. Red and photographic infrared linear combinations for monitoring vegetation. Remote sensing of Environment, 8: 127-150.

Examples

```
## Not run:
require(raster)

data(tab)

temp <-list()

# we create 4 simulated rasters for the data included in the object tab:

for(i in 1:4) {
temp[[i]] <- runif(19800, 0, 254)
temp[[i]] <- matrix(temp[[i]], 180, 110)
```

```
temp[[i]] <- raster(temp[[i]], crs="+proj=utm")
extent(temp[[i]])<-c(3770000, 3950000, 6810000, 6920000)
}

writeRaster(temp[[1]], "20040719b4.tif", overwrite=T)
writeRaster(temp[[2]], "20040719b3.tif", overwrite=T)
writeRaster(temp[[3]], "20091106b4.tif", overwrite=T)
writeRaster(temp[[4]], "20091106b3.tif", overwrite=T)

# Computing NDVI images:

eco.NDVI(tab, "COST", "NDVI", "LT5")

example <- raster("NDVICOST20040719.tif")
image(example)

## End(Not run)
```

`eco.NDVI.post` *Postprocessing for NDVI and MSAVI 2 temporal series of Landsat 5 and 7*

Description

This program must be used sequentially after `eco.NDVI`. The inputs required (tab, correct, method) are the same described and used in that function. The algorithm stacks the images and save the stack into the working directory with the name "time.tif". If the user wishes, the program can also compute images of max, min, mean and var by pixel over the temporal sequence. Default is "mean".

Usage

```
eco.NDVI.post(tab, correct = c("COST", "DOS"), method = c("NDVI", "MSAVI2"),
  datatype = c("FLT4S", "FLT8S", "INT4U", "INT4S", "INT2U", "INT2S", "INT1U",
  "INT1S", "LOG1S"), what = c("mean", "max", "min", "var", "none"))
```

Arguments

`tab`	Table used with `eco.NDVI`.
`correct`	Correction method used in `eco.NDVI`.
`method`	The vegetation index used in `eco.NDVI`.
`datatype`	Type of data, see `dataType`. Default "FLT4S".
`what`	Functions to apply over the created stack. The values permitted are: "none", "max", "min", "mean" and "var". The functions are implemented with `calc`. If more that one function would be applied, must be used the following syntax: c("fun_1", "fun:2", "fun_i").

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

References

Chander G., B. Markham, and D. Helder. 2009. Summary of current radiometric calibration coefficients for Landsat MSS, TM, ETM+, and EO-1 ALI sensors. Remote sensing of environment, 113: 893-903.

Chavez P. 1989. Radiometric calibration of Landsat Thematic Mapper multispectral images. Photogrammetric Engineering and Remote Sensing, 55: 1285-1294.

Chavez P. 1996. Image-based atmospheric corrections-revisited and improved. Photogrammetric engineering and remote sensing, 62: 1025-1035.

Goslee S. 2011. Analyzing remote sensing data in R: the landsat package. Journal of Statistical Software, 43: 1-25.

Song C., C. Woodcock, K. Seto, M. Lenney and S. Macomber. 2001. Classification and change detection using Landsat TM data: when and how to correct atmospheric effects?. Remote sensing of Environment, 75: 230-244.

Tucker C. 1979. Red and photographic infrared linear combinations for monitoring vegetation. Remote sensing of Environment, 8: 127-150.

See Also

eco.NDVI

extract

Examples

```
## Not run:
require(raster)

data(tab)
data(eco3)
temp <- list()

# we create 4 simulated rasters for the data included in the object tab:

for(i in 1:4) {
temp[[i]] <- runif(19800, 0, 254)
temp[[i]] <- matrix(temp[[i]], 180, 110)
temp[[i]] <- raster(temp[[i]], crs="+proj=utm")
extent(temp[[i]])<-c(3770000, 3950000, 6810000, 6920000)
}

writeRaster(temp[[1]], "20040719b4.tif", overwrite = T)
writeRaster(temp[[2]], "20040719b3.tif", overwrite = T)
writeRaster(temp[[3]], "20091106b4.tif", overwrite = T)
writeRaster(temp[[4]], "20091106b3.tif", overwrite = T)
```

```
# Computing NDVI images:

eco.NDVI(tab, "COST", "NDVI", "LT5")

# Mean NDVI image computed over the NDVI images that we calculated:

eco.NDVI.post(tab, "COST", "NDVI", what = c("mean", "var"))
mean.ndvi <- raster("NDVI.COST.mean.tif")
plot(mean.ndvi)

# Extraction of the mean NDVI for each point in the object eco and plot
# of the data:

ndvi <- extract(mean.ndvi, eco3[["XY"]])
ndvi<- aue.rescale(ndvi)
plot(eco3[["XY"]][, 1], eco3[["XY"]][, 2], col=rgb(ndvi, 0, 0),
pch=15, main = "Mean NDVI", xlab = "X", ylab  = "Y")

## End(Not run)
```

`eco.order` *Ordering the rows of the data frames contained in an ecogen object*

Description

Ordering the rows of the data frames contained in an ecogen object

Usage

```
eco.order(eco)
```

Arguments

`eco`	Object of class "ecogen".

Details

This program generates an ecogen object with the rows of all the data frames ordered in reference to the row names of the XY data frame. This is useful when the data frames are loaded into the ecogen object, but were not ordered previously. Also, this tool can be useful for reorder rows when is needed. First, the reference data frame in the slot XY will be in the desired row order. This program then aligns all the data frames by coincidence of row names with those in the slot XY.

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

Examples

```
## Not run:

data(eco.test)
eco1 <- eco
eco1[["P"]] <- eco[["P"]][sample(1:225), ]  #object with shuffled rows
eco1[["E"]] <- eco[["E"]][sample(1:225), ]
ordered <- eco.order(eco1)
head(ordered[["P"]]); head(eco[["P"]])


## End(Not run)
```

`eco.pairtest` *Kruskall - Wallis + Wilcoxon (Mann-Whitney U) and aov + Tukey-HSD tests for an ecogen object*

Description

Kruskall - Wallis + Wilcoxon (Mann-Whitney U) and aov + Tukey-HSD tests for an ecogen object

Usage

```
eco.pairtest(eco, df = c("P", "E", "A", "C"), x, test = c("wilcoxon",
  "tukey"), adjust = "fdr", only.p = TRUE, ...)
```

Arguments

`eco`	Object of class "ecogen".
`df`	The data frame for the analysis. Could be "P", "E" or "C".
`x`	The name of the S slot column with the groups for the analysis.
`test`	Test to perform ("wilcoxon", "tukey").
`adjust`	P-values correction method for multiple tests passed to `p.adjust`. Defalut is "fdr".
`only.p`	Should be only returned a matrix with P-values? Default TRUE.
`...`	Additional arguments passed to `wilcox.test` or `TukeyHSD`.

Details

This program returns the Wilcoxon (Mann-Whitney U) or Tukey-HSD statistics and p values for the multiple comparisons of the variables contained in the selected data frame, among the levels of a factor of the slot "S".

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

See Also

wilcox.test TukeyHSD

Examples

```
## Not run:
data(eco3)
wil <- eco.pairtest(eco = eco3, df = "P", x = "structure")
wil
wil <- eco.pairtest(eco = eco3,df = "E", x = "structure")
wil
wil <- eco.pairtest(eco = eco3, df = "P", x = "structure", only.p = FALSE)
wil
wil <- eco.pairtest(eco = eco3,df = "P", x = "structure", test = "tukey")
wil

## End(Not run)
```

---

eco.post.geneland *Log posterior probability plot for Geneland repetitions with fixed K*

Description

Log posterior probability plot for Geneland repetitions with fixed K

Usage

```
eco.post.geneland(niter, burnin)
```

Arguments

niter	Number of mcmc iterations per repetition.
burnin	Number of mcmc to burn-in.

Details

This program returns, for a series of Geneland repetitions with fixed K, and a specified burn-in value, a plot of the log posterior probability vs the repetition number. This allow to choose the best run. The working directory will be setted to the folder containing the results created by Geneland. The program expects each subfolder (run) having a number as name, indicating the corresponding number of run. (1, 2, etc., see the example).

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

Examples

```
## Not run:
require("Geneland")
data(eco.test)

# We create a folder in the working directory for the results and
# save the data frames of the object "eco" in the format required
# by Geneland:

path.1 <- getwd()
path <- paste(path.1,"/test/", sep="")
dir.create(path)
setwd(path)
eco.2geneland(eco, ploidy = 2)

# Auxiliar function for running some repetitions with fixed K = 4.
Each repetition is saved in the folder "test":
simul <- function(i) {
path <- getwd()
path <- paste(path,"/",i, sep = "")
dir.create(path)
MCMC(coordinates = read.table("XY.txt"),
 geno.dip.codom = read.table("G.txt"),
 varnpop = TRUE, npopmin = 4, npopmax = 4, spatial = TRUE,
 freq.model = "Correlated", nit = 500, thinning = 10,
 path.mcmc = path)
}

# 5 repetitions with K = 4
lapply(1:5, simul)

#Check that in the folder "test" are the simulated result.
Your results must have that appearance.

# Plot of the repetition order number vs the corresponding
# posterior probability, with a burn-in of 10 mcmc:
eco.post.geneland(5, 10)

## End(Not run)
```

`eco.rankplot` *Rankplot graphs*

Description

This function generates a plot for a numeric or factor variable. A data frame/matrix with XY coordinates is required. The X and Y axes in the plot correspond to the rank of the X and Y coordinates, respectively.

Usage

```
eco.rankplot(input, XY, xlabel = NULL, ylabel = NULL, title = NULL,
  legendlabel = NULL, background = c("grey", "white"), ...)

## S4 method for signature 'eco.lsa,missing,missing'
eco.rankplot(input, XY, xlabel, ylabel,
  title, legendlabel, background = c("grey", "white"), significant = TRUE,
  ns = NULL)

## S4 method for signature 'numeric,dataframeORmatrix,missing'
eco.rankplot(input, XY, xlabel,
  ylabel, title, legendlabel, background = c("grey", "white"))

## S4 method for signature 'factor,dataframeORmatrix,missing'
eco.rankplot(input, XY, xlabel,
  ylabel, title, legendlabel, background = c("grey", "white"))
```

Arguments

input	Numeric/factor variable.
XY	Data frame or matrix with X-Y coordinates.
xlabel	Optional label for x axis.
ylabel	Optional label for y axis.
title	Optional title label.
legendlabel	Optional legend label.
background	color of the background ("grey" or "white")-
...	Additional elements to the generic.
significant	should be colored only the individuals with significant result?. This argument can be used with eco.lsa results. Default TRUE
ns	color for non significant individuals, when significant = TRUE. This argument can be used with eco.lsa results.

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

Examples

```
## Not run:
data(eco3)

# The data set eco3 has  50 points in two sites,
# but they are not visible in a usual X-Y plot
due to the small distance among them

var <- eco3[["P"]][,1]
plot(eco3[["XY"]], col = var)
```

```
x <- sample(1:100, 30)
y <- sample(1:100, 30)

# in a rankplot graph, the inter-individual distances are
# reduced to a single scale
rankeco3 <- eco.rankplot(var, eco3[["XY"]])
rankeco3

# the rankplot method support the use of ggplot2 syntax
rankeco3 <- rankeco3 + theme_bw() + theme(legend.position="none")
rankeco3

## End(Not run)
```

`eco.rbind` *Combining the rows of two ecogen objects*

Description

Combining the rows of two ecogen objects

Usage

```
eco.rbind(eco1, eco2, ..., check.col.names = TRUE)
```

Arguments

`eco1`	Object of class "ecogen".
`eco2`	Object of class "ecogen".
`...`	Other "ecogen" objects to combine.
`check.col.names`	Check for duplicated column names? Default TRUE.

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

Examples

```
## Not run:

data(eco.test)

# duplicated row names are not allowed by eco.rbind.

eco2 <- eco

# row names not duplicated for P
```

```
rownames(eco2[["P"]]) <-226:450

eco.r <- eco.rbind(eco, eco2)

eco.r


## End(Not run)
```

`eco.remove` *Creating an updated ecogen object by removing results of the slot OUT*

Description

Creating an updated ecogen object by removing results of the slot OUT

Usage

```
eco.remove(eco, ...)
```

Arguments

`eco`	Object of class "ecogen".
`...`	Objects to remove from eco, typed without quotations.

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

Examples

```
## Not run:

data(eco.test)
variog <- eco.variogram(eco[["P"]][, 1], eco[["XY"]])

# Assignation of values can be made with the corresponding accessors,
# using the generic notation of EcoGenetics
# (<ecoslot.> + <name of the slot> + <name of the object>).
# See help("EcoGenetics accessors")

ecoslot.OUT(eco) <- variog
we.are.numbers <- c(1:10)
we.are.characters <- c("John Coltrane", "Charlie Parker")
ecoslot.OUT(eco) <- list(we.are.numbers, we.are.characters)
ecoslot.OUT(eco)
eco <- eco.remove(eco, we.are.numbers)
ecoslot.OUT(eco)


## End(Not run)
```

eco.subset *Subsetting an ecogen object by group*

Description

Subsetting an ecogen object by group

Usage

```
eco.subset(eco, fact, grp, missing = c("0", "NA", "MEAN"))
```

Arguments

eco	Object of class "ecogen".
fact	The name of the S slot column with labels assigning individuals to groups.
grp	Label for the subset of individuals, contained in fact.
missing	Missing data argument This can take three values ("0", "NA" or "MEAN"), as described in ecogen. Missing elements are treated as zeros in the default option.

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

Examples

```
## Not run:
data(eco3)
eco3
eco.sub <-eco.subset(eco3,"structure", 1)
eco.sub

## End(Not run)
```

eco.theilsen *Theil-sen regression for a raster time series*

Description

This function computes the theil-sen estimator and the P-value associated for each pixel over time in a stack of images, writing the values in a raster (one for the estimators and one for the P-values). It is recommended to use a "RasterBrick", that is more efficient in managing memory.

Usage

```
eco.theilsen(stacked, date, adjust = "none")
```

Arguments

stacked	Stacked images ("RasterLayer" or "RasterBrick").
date	data vector with dates for each image.
adjust	P-values correction method for multiple tests passed to p.adjust. Defalut is "none".

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

References

Sen, P. 1968. Estimates of the regression coefficient based on Kendall's tau. Journal of the American Statistical Association, Taylor and Francis Group, 63: 1379-1389.

Theil H. 1950. A rank-invariant method of linear and polynomial regression analysis, Part 3 Proceedings of Koninalijke Nederlandse Akademie van Weinenschatpen A, 53: 397-1412.

See Also

rkt.

Examples

```
## Not run:
require("raster")
set.seed(6)

temp <- list()
for(i in 1:100) {
temp[[i]] <- runif(36,-1, 1)
temp[[i]] <- matrix(temp[[i]], 6, 6)
temp[[i]] <- raster(temp[[i]])
}

temp <- brick(temp)


writeRaster(temp,"temporal.tif", overwrite=T)
rm(temp)
ndvisim <- brick("temporal.tif")

date <- seq(from = 1990.1, length.out = 100, by = 0.2)

eco.theilsen(ndvisim, date)

pvalue <- raster("pvalue.tif")
slope <- raster("slope.tif")
par(mfrow = c(1, 2))
plot(pvalue, main = "p-value")
plot(slope, main = "slope")
```

```
## End(Not run)
```

`eco.variogram` *Empirical variogram*

Description

This program computes the empirical variogram of a selected variable. If the coordinates are given in decimal degrees, set latlon = TRUE. The program return a table with the mean class distances (d.mean) and the semivariances (obs) for each class.

Usage

```
eco.variogram(Z, XY, int = NULL, smin = 0, smax = NULL, nclass = NULL,
  seqvec = NULL, size = NULL, bin = c("sturges", "FD"), row.sd = FALSE,
  latlon = FALSE)
```

Arguments

`Z`	Vector for the analysis.
`XY`	Data frame or matrix with individual's position (projected coordinates).
`int`	Distance interval in the units of XY.
`smin`	Minimum class distance in the units of XY.
`smax`	Maximum class distance iin the units of XY.
`nclass`	Number of classes.
`seqvec`	Vector with breaks in the units of XY.
`size`	Number of individuals per class.
`bin`	Rule for constructing intervals when a partition parameter (int, nclass or size) is not given. Default is Sturge's rule (Sturges, 1926). Other option is Freedman-Diaconis method (Freedman and Diaconis, 1981).
`row.sd`	Logical. Should be row standardized the matrix? Default FALSE (binary weights).
`latlon`	Are the coordinates in decimal degrees format? Defalut FALSE. If TRUE, the coordinates must be in a matrix/data frame with the longitude in the first column and latitude in the second. The position is projected onto a plane in meters with the function `geoXY`.

Value

The program returns an object of class "eco.correlog" with the following slots:

> OUT analysis output

> IN analysis input data

> BEAKS breaks

> CARDINAL number of elements in each class

> DISTMETHOD method used in the construction of breaks

ACCESS TO THE SLOTS The content of the slots can be accessed with the corresponding accessors, using the generic notation of EcoGenetics (<ecoslot.> + <name of the slot> + <name of the object>). See help("EcoGenetics accessors") and the Examples section below

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

References

Borcard D., F. Gillet, and P. Legendre. 2011. Numerical ecology with R. Springer Science & Business Media.

Legendre P., and L. Legendre. 2012. Numerical ecology. Third English edition. Elsevier Science, Amsterdam, Netherlands.

Examples

```
## Not run:

data(eco.test)
variog <- eco.variogram(Z = eco[["P"]][, 2],XY =  eco[["XY"]])
plot(variog)

# variogram plots support the use of ggplot2 syntax
variogplot <- plot(variog) + theme_bw() + theme(legend.position="none")
variogplot

#-----------------------
# ACCESSORS USE EXAMPLE
#-----------------------

# the slots are accessed with the generic format
# (ecoslot. + name of the slot + name of the object).
# See help("EcoGenetics accessors")

ecoslot.OUT(variog)        # slot OUT
ecoslot.BREAKS(variog)     # slot BREAKS


## End(Not run)
```

---

eco.weight *Spatial weights*

Description

Spatial weights for individuals with coordinates XY

Usage

```
eco.weight(XY, method = c("circle", "knearest", "inverse", "circle.inverse",
  "exponential", "circle.exponential"), d1 = 0, d2 = NULL, k = NULL,
  p = 1, alpha = 1, dist.method = "euclidean", row.sd = FALSE,
  self = FALSE, latlon = FALSE)
```

Arguments

XY	Matrix/data frame with projected coordinates.
method	Method of spatial weight matrix: "circle", "knearest", "inverse", "circle.inverse", "exponential", "circle.exponential".
d1	Minimum distance for circle matrices.
d2	Maximum distance for circle matrices.
k	Number of neighbors for nearest neighbor distance. When equidistant neighbors are present, the program select them randomly.
p	Power for inverse distance. Default = 1.
alpha	Alpha value for exponential distance. Default = 1.
dist.method	Method of computing distance when XY is in metric units. If latlon is TRUE, the method is euclidean.
row.sd	Logical. Should be row standardized the matrix? Default FALSE (binary weights).
self	Should be the individuals self-included in circle or knearest weights? Defalut FALSE.
latlon	Are the coordinates in decimal degrees format? Defalut FALSE. If TRUE, the coordinates must be in a matrix/data frame with the longitude in the first column and latitude in the second. The position is projected onto a plane in meters with the function geoXY.

Details

This program computes a weights matrix (square matrix with individuals in rows and columns, and weights wij in cells (i and j, individuals)) under the following available methodologies:

- circle: all the connection between individuals i and j, included in a distance radius, higher than d1 and lower than d2, with center in the individual i, have a value of 1 for binary weights. This distance requires the parameters d1 and d2 (default d1 = 0).

- knearest: the connections between an individual and its nearest neighbors of each individual i have a value of 1 for binary weights. This distance requires the parameter k.

- inverse: inverse distance with exponent p (distance = 1/dij^p, with dij the distance between individuals i and j). This distance requires the parameter p (default p = 1).

- circle inverse: combination of "circle" and "inverse". It is the matrix obtained by multiplying each element in a "circle" binary matrix, and an "inverse" matrix. This distance requires the parameters p, d1 and d2 (default p = 1, d1 = 0).

- exponential: inverse exponential distance with parameter alpha (distance = 1/e^(alpha *dij), with dij the distance between individuals i and j). This distance requires the parameter alpha (default alpha = 1).

- circle exponential: combination of "circle" and "exponential". It is the matrix obtained by multiplying each element in a "circle" binary matrix, and an "exponential" matrix. This distance requires the parameters alpha, d1 and d2 (default alpha = 1, d1 = 0).

In row standardization, each weight wij for the individual i, is divided by the sum of the row weights (i.e., wij / sum(wij), where sum(wij) is computed over an individual i and all individuals j).

When self is TRUE, the connection j = i is also included.

Value

An object of class eco.weight with the following slots:

> W weights matrix

> XY input coordinates

> METHOD weights construction method

> PAR parameters used for the construction of weights

> PAR.VAL values of the parameters used for the construction of weights

> ROW.SD row standardization (logical)

> SELF data self-included (logical)

> NONZERO percentage of non-zero connections

> NONZEROIND percentage of individuals with non-zero connections

> AVERAGE average number of connection per individual

ACCESS TO THE SLOTS The content of the slots can be accessed with the corresponding accessors, using the generic notation of EcoGenetics (<ecoslot.> + <name of the slot> + <name of the object>). See help("EcoGenetics accessors") and the Examples section below

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

Examples

```
## Not run:

data(eco.test)

# "circle" method
con <- eco.weight(eco[["XY"]], method = "circle", d1 = 0, d2 = 2)
con

# "knearest" method
con <- eco.weight(eco[["XY"]], method = "knearest", k=3)
con

# "inverse" method
con <- eco.weight(eco[["XY"]], method = "inverse")
con
```

```
# "circle.inverse" method
con <- eco.weight(eco[["XY"]], method = "circle.inverse", d2 = 2)
con

# "exponential" method
con <- eco.weight(eco[["XY"]], method = "exponential")
con

# "circle.exponential" method
con <- eco.weight(eco[["XY"]], method = "circle.exponential", d2 = 2)
con


#-----------------------
# ACCESSORS USE EXAMPLE
#-----------------------

# the slots are accessed with the generic format
# (ecoslot. + name of the slot + name of the object).
# See help("EcoGenetics accessors")

ecoslot.METHOD(con)        # slot METHOD
ecoslot.PAR(con)           # slot PAR
ecoslot.PAR.VAL(con)       # slot PAR.VAL


## End(Not run)
```

---

eco2 *Eco2*

Description

ecogen object with simulated data of 900 individuals.

Usage

```
data(eco2)
eco2
```

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

eco3 *Eco3*

Description

ecogen object with simulated data of 173 individuals.

Usage

```
data(eco3)
eco3
```

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

ecogen *Creating a new ecogen object*

Description

Creating a new ecogen object

Usage

```
ecogen(XY = data.frame(), P = data.frame(), G = data.frame(),
  E = data.frame(), S = data.frame(), C = data.frame(),
  G.processed = TRUE, order.G = FALSE, type = c("codominant", "dominant"),
  ploidy = 2, sep, ncod = NULL, missing = c("0", "NA", "MEAN"),
  NA.char = "NA", poly.level = 5, rm.empty.ind = FALSE)
```

Arguments

XY	Data frame with m columns (coordinates) and n rows (individuals).
P	Data frame with n rows (individuals), and phenotypic data in columns.
G	Data of class: "data.frame", with individuals in rows and genotypic data in columns (loci). The ploidy and the type (codominant, dominant) of the data, must be passed with the arguments "ploidy" and "type". Missing data is coded as NA. Dominant data must be coded with binary values (0 for absence - 1 for presence).
E	Data frame with n rows (individuals), and environmental data in columns.
S	Data frame with n rows (individuals), and groups (factors) in columns. The program converts non factor data into factor.
C	data frame with n rows (individuals), and custom variables in columns.

G.processed	If TRUE, the slot G will include a processed data frame (removed non informative loci (the data non available for all the individuals), removed non polymorphic loci (for dominant data) and ordered alleles in ascending order.
order.G	Genotypes must be ordered in G slot? (codominant data) Default FALSE.
type	Marker type: "codominant" or "dominant".
ploidy	Ploidy of the G data frame. Default ploidy = 2.
sep	Character separating alleles (codominant data). Default option is no character separating alleles.
ncod	Number of characters coding each allele (codominant data).
missing	Missing data treatment ("0", "NA", or "MEAN") for the A slot. Missing elements are set to 0 in the default option. missing elements are recoded as "NA" or the mean allelic frequency across individuals in "NA" and "MEAN" options, respectively.
NA.char	Character simbolizing missing data in the input. Default is "NA".
poly.level	Polymorphism threshold in percentage (0 - 100), for remotion of non polymorphic loci (for dominant data). Default is 5 (5%).
rm.empty.ind	Remotion of noninformative individuals (row of "NAs"). Default if FALSE.

Details

This is a generic function for creating an ecogen object. In default option, the missing data input value is "NA", but any missing data character can be passed with the option NA.char. The output in the slot G will have coded the missing data as NA.

ACCESS TO THE SLOTS. MODIFICATION OF ECOGEN OBJECTS

The content of the slots can be extracted with the corresponding accesors ecoslot.XY, ecoslot.P, ecoslot.G, ecoslot.A, ecoslot.E, ecoslot.C and ecoslot.OUT. Data can be assigned individually to the slots, also with the corresponding accessors. The correct use of ecogen objects requires the implementation of accessors, as they ensure the data check and pre-processing. The use of accessors enables to modify or fill the slots of ecogen objects, without the need of creating a new object each time. See *help("EcoGenetics accessors")* for a detailed description and examples about ecogen accessors.

OTHER SLOT ACCESS METHODS FOR ECOGEN OBJECTS

The use of brackets is defined for ecogen objects:

- Single bracket: the single bracket ("[") is used to subset all the ecogen data frames (P, G, E, S, A and C) by row, at once. The notation for an object is eco[from:to], where eco is any ecogen object, and from: to is the row subset. For example: eco[1:10] , subsets the object eco from row 1 to row 10, for all the data frames at once.

- Double square brackets: the double square brackets are symbolic abbreviations of the accessors (i.e., it is a call to the corresponding accessor). The usage is: eco[["X"]], where X is any slot of interest: eco[["P"]], eco[["G"]], eco[["A"]], eco[["E"]], eco[["S"]], eco[["C"]] and eco[["OUT"]]. Double square brackets can be used in get/set mode. See Examples below and in help("EcoGenetics accessors").

ABOUT THE CONSTRUCTION OF NEW ECOGEN OBJECTS

The construction of a new ecogen object can be made in two different ways. First, a new object can be created, incorporating all the information at once. Second, the data can be added in each slot, using the corresponding accessor / "[[". The accessor/double square brackets methods allow temporal modification of any created ecogen object and ensures its modularity. These methods are not only functions to get/assign values to the slots, they provide a basic pre-processing of the data during assignment, generating a set of coherent information.

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

Examples

```
## Not run:

#Example with G data of class "data.frame", corresponding to
#microsatellites of a diploid organism:
data(eco.test)
eco <- ecogen(XY = coordinates, P = phenotype, G = genotype,
E = environment, S = structure)

#Example with G data of class "data.frame", corresponding to a
#presence - absence molecular marker:
dat <- sample(c(0,1),100,rep = TRUE)
dat <- data.frame(matrix(dat,10,10))
eco <- ecogen(G = dat, type = "dominant")


# DINAMIC ASSIGNMENT WITH ACCESSORS AND "[["

eco <- ecogen(XY = coordinates, P = phenotype)
eco

ecoslot.G(eco, order.G = TRUE) <- genotype

# this is identical to
eco[["G", order.G=TRUE]] <- genotype

ecoslot.E(eco) <- environment

# this is identical to
eco[["E"]] <- environment

#-----------------------------------------------------------
# See additional examples in help("EcoGenetics accessors")
#-----------------------------------------------------------

# Storing data in the slot OUT

 singers <- c("carlos_gardel", "billie_holiday")

ecoslot.OUT(eco) <- singers
```

```
# Storing several data

golden.number <- (sqrt(5) + 1) / 2
ecoslot.OUT(eco) <- list(singers, golden.number)   # several objects must be passed as a list

# this is identical to:

eco[["OUT"]] <- list(singers, golden.number)


## End(Not run)
```

`environment` *environment*

Description

Data frame with simulated environmental variables of 225 individuals.

Usage

```
data(eco.test)
environment
```

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

`genotype` *genotype*

Description

Data frame with simulated microsatellite data of 225 individuals.

Usage

```
data(eco.test)
genotype
```

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

int.loc2al *INTERNAL CONVERSION TOOLS FOR GENETIC DATA*

Description

INTERNAL CONVERSION TOOLS FOR GENETIC DATA

Usage

```
int.loc2al(X, ncod = NULL, ploidy = 2, sep.in = "", sep.out = NULL,
  chk.names = TRUE, chk.plocod = TRUE)
```

Arguments

X	Input
ncod	Number of digits coding each allele (e.g., 1: x, 2: xx, 3: xxx, etc.). If NULL, ncod will we obtained from the ploidy and the maximum number of characters in the data cells.
ploidy	Ploidy of the data.
sep.in	Separator in the input
sep.out	Separator in the output
chk.names	Defalult TRUE. The function make checks of individuals and loci names during conversion.
chk.plocod	Defalult TRUE. The function checks coherence in ploidy and number of digits coding alleles for loci data during conversion.

phenotype *phenotype*

Description

Data frame with simulated morphometric data of 225 individuals.

Usage

```
data(eco.test)
phenotype
```

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

structure *structure*

Description

Factor with simulated groups of 225 individuals.

Usage

```
data(eco.test)
structure
```

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

summary,eco.mlm-method *Summary for eco.lmtree output*

Description

Summary for eco.lmtree output

Usage

```
## S4 method for signature 'eco.mlm'
summary(object)

## S4 method for signature 'eco.mctree'
summary(object)
```

Arguments

object	Output object of eco.lmtree.

Value

A Table with a summary of the analysis for "mlm" analysis, the plot of the trees with significant splits for "mctree" analysis.

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

See Also

eco.lmtree

Examples

```
## Not run:

data(eco.test)
#' mod <- eco.lmtree(DF1 = eco$P, DF2 = eco$E,
analysis = "mlm")
summary(mod)                                    #summary for "mlm" analysis

mod <- eco.lmtree(DF1 = eco$P, DF2 = eco$E,
analysis = "mctree", fact = eco$S$structure)
summary(mod)                                    #summary for "mctree" analysis


## End(Not run)
```

---

tab *tab*

Description

Data frame with information of bands 3 and 4 for two dates, corresponding to real Landsat 5 images and used in this package as pedagogic material complementing simulated data. Date and sun elevation data were extracted from the header provided with the image in http://glovis.usgs.gov/. The starting haze values (SHV) were estimated checking the profiles of the bands.

Usage

```
data(tab)
tab
```

Author(s)

Leandro Roser <leandroroser@ege.fcen.uba.ar>

Index