Package ‘pgirmess’

March 12, 2018

Version 1.6.9

Date 2018-03-11

Title Spatial Analysis and Data Mining for Field Ecologists

Author Patrick Giraudoux [aut, cre],
Jean-Philippe Antonietti [ctb],
Colin Beale [ctb],
David Pleydell [ctb],
Mike Treglia [ctb]

Maintainer Patrick Giraudoux <patrick.giraudoux@univ-fcomte.fr>

Description Set of tools for reading, writing and transforming spatial and seasonal data in ecology, model selection and specific statistical tests. It includes functions to discretize polylines into regular point intervals, link observations to those points, compute geographical coordinates at regular intervals between waypoints, read subsets of big rasters, compute zonal statistics or table of categories within polygons or circular buffers from raster. The package also provides miscellaneous functions for model selection, spatial statistics, geometries, writing data.frame with Chinese characters, and some other functions for field ecologists.

Imports boot (>= 1.3-4), maptools (>= 0.8-36), rgdal (>= 0.7-8), rgeos (>= 0.3-8), sp (>= 0.9-97), spdep (>= 0.5-43), splancs (>= 2.01-31)

Suggests MASS (>= 7.3-1), nlme (>= 3.1-120)

License GPL (>= 2)

URL http://perso.orange.fr/giraudoux

NeedsCompilation no

Repository CRAN

Date/Publication 2018-03-12 14:56:20 UTC

R topics documented:

`CI` *Confidence interval of percentages*

Description

Computes the lower limit and upper limit of the 95 percent confidence interval of percentage estimates

Usage

```
CI(x, ...)
```

Arguments

`x`	a two-dimensional table, matrix or data.frame with 2 columns, giving the counts of successes and failures, respectively
`...`	other arguments to pass to `prop.test`, eg `conf.level`

Details

Simple wrapper of `prop.test`. The default confidence interval is 95 percent, but can be modified passing values to `prop.test` by the `conf.level` argument.

Value

A 3 column matrix.

- Column 1: percentage estimate
- Column 2: lower limit of the confidence interval
- column 3: upper limit of the confidence interval

See Also

`prop.test`

Examples

```
x<-c(2,10,7,8,7) # eg: number of positive cases
y<-c(56,22,7,20,5)# eg: number of negative cases
CI(cbind(x,y))
CI(cbind(x,y), conf.level=0.99)
```

---

classnum *Gives an index vector of the class category of each value of a numerical vector*

Description

Gives an index vector of the class category of each value of a numerical vector

Usage

```
classnum(x, breaks = "Sturges")
```

Arguments

`x`	a vector of values for which the indices are desired
`breaks`	one of: • a vector giving the breakpoints between bins, • a single number giving the number of bins, • a character string naming an algorithm to compute the number of cells (see Details).

Details

The default for 'breaks' is '"Sturges"': see 'nclass.Sturges'. Other names for which algorithms are supplied are '"Scott"' and '"FD"' for '"Friedman-Diaconis"' (with corresponding functions 'nclass.scott' and 'nclass.FD'). Case is ignored and partial matching is used. Breaks and labels are stored as attributes.

Value

A vector of the same length as x, with the index of the class which each value of x belongs to

See Also

cut, classIntervals

Examples

```
x<-rnorm(30)
classnum(x)
classnum(x,breaks="fd")
classnum(x, breaks=c(-1,0,1))
classnum(x,breaks=5)
```

---

cormat — *Gives a correlation matrix and the probability of Ho for each correlation*

Description

Gives a correlation matrix and the probability of Ho for each correlation estimate

Usage

```
cormat(donnees, method = "spearman", sep = FALSE)
```

Arguments

`donnees`	a data frame of numerics
`method`	a string of characters among 'pearson', 'spearman' (default), 'kendall'
`sep`	If true, gives the results in two matrices (default = F)

Details

Wrapper for 'cor' and 'cor.test'. The results can be given in one or two matrices.

Value

If sep = F (default) a list including:

`method`	The method used
`prob.cor`	Upper triangle, the correlations; lower triangle, the probability of Ho

If sep = T a list including:

`method`	The method used
`coef.estimates`	The correlation matrix
`p.value`	The Ho probability matrix

See Also

`cor`, `cor.test`

Examples

```
cormat(longley)
cormat(longley,sep=TRUE)
```

---

correlog — *Computes Moran's or Geary's coefficients on distance classes*

Description

Computes Moran's or Geary's coefficients on distance classes from a set of spatial coordinates and corresponding z values

Usage

```
correlog(coords, z, method="Moran", nbclass = NULL,...)
```

Arguments

`coords`	a two columns array, data.frame or matrix of spatial coordinates. Column 1 = X, Column 2 = Y.
`z`	a vector for the values at each location. Must have the same length as the row number of coords
`method`	the method used. Must be "Moran" (default) or "Geary"
`nbclass`	number of bins. If NULL Sturges method is used to compute an optimal number
`...`	further arguments to pass to e.g. `moran.test` or `geary.test`

Details

Uses the library spdep including `moran.test` or `geary.test`. Distances are euclidian and in the same unit as the spatial coordinates. Moran's Ho: I values larger than 0 due to chance; Geary's Ho: C values lesser than 1 due to chance. Correlog has print and plot methods; statistically significant values (p<0.05) are plotted in red.

Value

An object of class "correlog", a matrix including:

`class`	bin centers
`I`	the coefficient values
`p.value`	probability of Ho
`n`	the number of pairs

Warning

Computing can take a long time for large data sets

Author(s)

Patrick Giraudoux pgiraudo@univ-fcomte.fr, Colin Beale c.beale@macaulay.ac.uk and Mike Treglia mike-treglia@utulsa.edu

References

see library spdep

See Also

`geary.test`, `moran.test`

Examples

```
library(spdep)
data(oldcol)
attach(COL.OLD)
coords<-cbind(X,Y)
res<-correlog(coords,CRIME)
plot(res)

res<-correlog(coords,CRIME,method="Geary")
plot(res)
```

`date2winter` *Convert a POSIXt date into categories corresponding to a autumn/winter/spring sequence*

Description

Convert a POSIXt date into categories corresponding to the time spanning from the late months of a year to the early months of the following year

Usage

```
date2winter(x, first = 10, last=4)
```

Arguments

`x`	a vector of POSIXt dates
`first`	number of the first month to include (default 10, October)
`last`	number of the last month to include (default 4, April)

Details

In ecology, time data must often be analysed on a time span category covering two successive years (e.g. the winter period). This function convert POSIXt dates into categories corresponding to the time span stretching from a user defined month of a given year (by default October) to a user-defined month of the following year (by default April). If date month is out of the user defined time span the value 'Excluded' is returned.

Value

A vector of the same length as x, with the time span category each value belongs to.

Examples

```
dates <- strptime(c("02/12/2002", "15/01/2003","15/10/2003","15/6/2003",NA),"%d/%m/%Y")
date2winter(dates)
```

diag2edge *Computes the edge of a square from its diagonal*

Description

Computes the edge of a square from its diagonal.

Usage

```
diag2edge(cordseg)
```

Arguments

cordseg	The diagonal coordinates. This can be a vector c(x1,y1,x2,y2), a 2 x 2 matrix or a data.frame (each line a coordinate)

Details

The first point coordinates are the left top of the diagonal. The other coordinates computed are the other top of the square edge. Can be used e.g. to pass a square edge to pave in order to compute a sampling grid.

Value

A 2x2 matrix of points coordinates

See Also

pave

Examples

```
# diagonal sloping up
coord<-matrix(c(20,20,90,90),nr=2,byrow=TRUE)
plot(coord,type="n",xlim=c(0,100),ylim=c(0,110),asp=1)
lines(coord,lty=2)
# square edge
lines(diag2edge(coord),col="red")

# diagonal sloping down
coord<-matrix(c(20,90,90,20),nr=2,byrow=TRUE)
plot(coord,type="n",xlim=c(0,100),ylim=c(0,110),asp=1)
lines(coord,lty=2)
# square edge
lines(diag2edge(coord),col="red")

# diagonal vertical
coord<-matrix(c(20,90,20,20),nr=2,byrow=TRUE)
plot(coord,type="n",xlim=c(0,100),ylim=c(0,110),asp=1)
lines(coord,lty=2)
# square edge
lines(diag2edge(coord),col="red")
```

`difshannonbio` *Empirical confidence interval of the bootstrap of the difference between two Shannon indices*

Description

Computes the empirical confidence interval of the bootstrap of the difference between two Shannon indices

Usage

```
difshannonbio(dat1, dat2, R = 1000, probs = c(0.025, 0.975))
```

Arguments

`dat1`	a data.frame of two columns; column = category, column 2 = biomass
`dat2`	a data.frame of two columns; column = category, column 2 = biomass
`R`	number of permutations
`probs`	the limits of the confidence interval

Details

Designated to compare the difference between two Shannon's indices computed from two data frames. In each data frame, the first column is the category of prey item, and the second column the estimated biomass.

Value

A list with the confidence interval of H' and J'

See Also

`shannonbio`

Examples

```
data(preybiom)
attach(preybiom)
jackal<-preybiom[site=="Y" & sp=="C",5:6]
genet<-preybiom[site=="Y" & sp=="G",5:6]

difshannonbio(jackal,genet,R=150)
```

---

dirProj — *Computes new coordinates given bearings and distances.*

Description

Computes new coordinates from bearings (North = 0) and distances

Usage

```
dirProj(df,deg=TRUE)
```

Arguments

`df`	a matrix or data frame of 4 columns giving x, y coordinates, bearings and distances
`deg`	if TRUE (default) bearings are in degree, otherwise in radian

Details

Computings are based on euclidian distance. Therefore, the coordinates should be given in a projected (plan) system (e.g. UTM, Lambert, etc.) and the distance in the same units as the projection system (e.g. meters).

Value

a matrix of two columns with the projected coordinates

See Also

`distSeg`, `gcDestination`

Examples

```
df<-data.frame(x1=0,y1=0,alpha=runif(3,0,360),d=runif(3,0,1))
df
plot(-1:1,-1:1,type="n")
points(0,0,pch=19)
points(dirProj(df))
text(dirProj(df)[,1],dirProj(df)[,2],1:3,pos=4)
```

`dirSeg` *Computes segment directions.*

Description

Computes the direction of segments from the first top clockwise (North = 0)

Usage

```
dirSeg(x,deg=TRUE)
```

Arguments

`x`	a matrix or data frame of 4 columns giving the coordinates of each segment tops x1, y1, x2, y2
`deg`	if TRUE (default) the output is in degrees, otherwise in radians

Details

The first two colomns give the first top coordinates, x then y, and the next two the second top coordinates.

Value

A vector of directions

See Also

`dirProj`, `gzAzimuth`

Examples

```
x2<-rnorm(10)
y2<-rnorm(10)
mydata<-cbind(0,0,x2,y2)
dirs<-dirSeg(mydata)
dirs
```

```
plot(range(mydata[,c(1,3)]),range(mydata[,c(2,4)]),type="n")
Segments(mydata)
text(mydata[,3],mydata[,4],paste(round(dirs,0),"\u00b0"),cex=0.7)
```

distNNeigh *Computes distances to the nearest neighbour*

Description

Computes distances to the nearest neighbour

Usage

```
distNNeigh(db)
```

Arguments

`db`	A matrix or data.frame of points coordinates column 1 = x,column 2 = y.

Details

Computes distances to the nearest neighbour for each line of a matrix of points coordinates

Value

A vector of distances

See Also

`knearneigh`, `knn2nb`, `nbdists`

Examples

```
distNNeigh(cbind(rnorm(30),rnorm(30)))
```

distNode	*Computes the distances between each nodes of a polyline.*

Description

Computes the distances between each nodes of a polyline.

Usage

```
distNode(pts,decdeg=FALSE)
```

Arguments

`pts`	A matrix or data.frame of the node coordinates column 1 = x,column 2 = y.
`decdeg`	TRUE if point coordinates are longitude-latitude decimal degrees, in which case distances are measured in meters

Details

If degdec is FALSE (default), distance computed is Euclidian. Units depends on the coordinate systems. If decdeg = TRUE, D = 1852 * 60 * (180/pi) * acos (sin(la1) * sin(la2) + cos(la1) * cos(la2) * cos(abs(lg1 - lg2)). This method calculates the great circle distance, is based on spherical trigonometry, and assumes that:

- 1 minute of arc is 1 nautical mile
- 1 nautical mile is 1.852 km

Value

A vector of distances

See Also

distTot, distSeg

Examples

```
x<-c(10,56,100)
y<-c(23,32,150)
distNode(cbind(x,y))
```

`distSeg` *Computes distances between the top coordinates of segments.*

Description

Computes the distances between the top coordinates of segments.

Usage

```
distSeg(mydata,decdeg=FALSE)
```

Arguments

`mydata`	A matrix or data frame of 4 columns giving the coordinates of each segment tops x1, y1, x2, y2
`decdeg`	TRUE if point coordinates are longitude-latitude decimal degrees, in which case distances are measured in meters

Details

If degdec is FALSE (default), distance computed is Euclidian. Units depends on the coordinate systems. If decdeg = TRUE, D = 1852 * 60 * (180/pi) * acos (sin(la1) * sin(la2) + cos(la1) * cos(la2) * cos(abs(lg1 - lg2)). This method calculates the great circle distance, is based on spherical trigonometry, and assumes that:

- 1 minute of arc is 1 nautical mile
- 1 nautical mile is 1.852 km

When computing with decdeg=TRUE duplicated coordinates strictly identical can lead to produce NaN. The corresponding distance is coerced to zero with warnings and if so, an attribute 'NaNcoerced2zero' with the row numbers of the distances that have been coerced to zero is created

Value

A vector of distances, possibly with the attribute 'NaNcoerced2zero' with the row numbers of the distances that have been coerced to zero if any.

See Also

`distNode`, `distTot`

Examples

```
x1<-rnorm(20)
y1<-rnorm(20)
x2<-rnorm(20)
y2<-rnorm(20)
mydata<-cbind(x1,y1,x2,y2)
distSeg(mydata)
```

distTot — *Computes the total length of a polyline.*

Description

Computes the total length of a polyline.

Usage

```
distTot(pts,decdeg=FALSE)
```

Arguments

`pts`	A matrix or data.frame of the node coordinates column 1 = x,column 2 = y.
`decdeg`	TRUE if point coordinates are longitude-latitude decimal degrees, in which case distances are measured in meters

Details

If degdec is FALSE (default), distance computed is Euclidian. Units depends on the coordinate systems. If decdeg = TRUE, D = 1852 * 60 * (180/pi) * acos (sin(la1) * sin(la2) + cos(la1) * cos(la2) * cos(abs(lg1 - lg2)). This method calculates the great circle distance, is based on spherical trigonometry, and assumes that:

- 1 minute of arc is 1 nautical mile
- 1 nautical mile is 1.852 km

Value

A numeric distance.

See Also

, `distNode`, `distSeg`

Examples

```
x<-c(10,56,100)
y<-c(23,32,150)
distTot(cbind(x,y))
```

expandpoly *Homothetia (size expansion) of a polygon*

Description

Compute the new coordinates of polygon expanded by a factor.

Usage

```
expandpoly(mypol, fact)
```

Arguments

`mypol`	matrix or data.frame of polygon coordinates
`fact`	expansion factor

Details

The polygon area obtained after expansion is equal to $fact^2$ times the original polygon area

Value

A matrix of polygon coordinates

See Also

`polygon`

Examples

```
x<-c(-5,-4.5,0,10,5)
y<-c(-10,0,5,5,-8)
poly<-cbind(x,y)
plot(-10:20,-20:10,type="n")
polygon(poly)
polygon(expandpoly(poly,1.5),border="red")
polygon(expandpoly(poly,0.5),border="blue")
```

findR — *Computes the distance between the centroid and the most distant coordinate of a geographical coordinate set*

Description

Computes the distance between the centroid and the most distant coordinate of a geographical coordinate set.

Usage

```
findR(coords)
```

Arguments

`coords`	A matrix or data frame of 2 columns of geographical coordinates

Value

The distance

See Also

`polycirc`

Examples

```
mydata<-cbind(x=rnorm(20),y=rnorm(20))
radius<-findR(mydata)
centroid<-matrix(colMeans(mydata),ncol=2)
plot(mydata,asp=1)
points(centroid,pch=19,col="red",cex=2)
polygon(polycirc(radius,centroid),border="red")
```

friedmanmc — *Multiple comparisons after Friedman test*

Description

Test of multiple comparison after Friedman test

Usage

```
friedmanmc(y, groups, blocks,probs=0.05)
```

Arguments

`y`	a numeric vector of data values, or a data matrix
`groups`	a vector giving the group for the corresponding elements of 'y' if this is a vector; ignored if 'y' is a matrix. If not a factor object, it is coerced to one.
`blocks`	a vector giving the block for the corresponding elements of 'y' if this is a vector; ignored if 'y' is a matrix. If not a factor object, it is coerced to one.
`probs`	a probability for the critical difference.

Details

Method for formula still not implemented. Formula 7.5a (Siegel & Castellan, 1988 p 180-181) can lead to p values larger than 1 when differences between groups are small. Eventually, they are set to NA and a warning is generated.

Value

A list of class 'mc' with the following items:

`statistic`	statistics used
`p.value`	the p value of the critical difference
`dif.com`	a data.frame with observed and critical differences

References

Siegel & Castellan (1988) Non parametric statistics for the behavioural sciences. Mc Graw Hill Int. Edt.

See Also

`friedman.test`

Examples

```
data(siegelp179)
attach(siegelp179)

friedman.test(score,treatment,block)
friedmanmc(score,treatment,block)
friedmanmc(score,treatment,block,probs=0.01)

mymatrix<-matrix(score,nc=3)
friedman.test(mymatrix)
friedmanmc(mymatrix)
detach(siegelp179)
```

gps2gpx — *Download waypoints or tracks from a GPS to a gpx file*

Description

Download waypoints or tracks from a GPS to a gpx file or to the console gpx formatted

Usage

```
gps2gpx(filename="",i="garmin",f = "usb:", type = "w", invisible = TRUE)
```

Arguments

`filename`	a character string naming the file to print to. If '""' (the default), prints to the standard output connection
`i`	INTYPE: a supported file type, default "garmin"
`f`	INFILE: the appropriate device interface, default "usb:", on Windows for serial interfaces commonly "com4:" or similar
`type`	"w" waypoints, or "t" track, or others provided in gpsbabel
`invisible`	Under Windows, do not open an extra window

Details

The function calls gpsbabel via the system. The gpsbabel program must be present and on the user's PATH for the function to work see `http://www.gpsbabel.org`. A .gpx suffix is added if not included in the filename. The gpx file can then be read e.g. using `readOGR` to a sp spatial object. Ex: readOGR("filename.gpx", "waypoints", drop_unsupported_fields=TRUE), or uploaded to a GPS

See Also

`readOGR`,`uploadGPS`

Examples

```
## Not run:
  # a GPS device must be connected
gps2gpx() # download waypoints and print to the console
gps2gpx(t="t") # download tracks and print to the console
gps2gpx(filename="myfile") # download waypoints and write a gpx file


## End(Not run)
```

kruskalmc *Multiple comparison test after Kruskal-Wallis*

Description

Multiple comparison test between treatments or treatments versus control after Kruskal-Wallis test

Usage

```
kruskalmc(resp,...)
## Default S3 method:
kruskalmc(resp, categ, probs = 0.05, cont=NULL,...)
## S3 method for class 'formula'
kruskalmc(resp,data=NULL,...)
```

Arguments

`resp`	a numeric vector of data values or a formula of the type 'response~category'.
`categ`	a factor object giving the group for the corresponding elements of 'x'
`probs`	the significance level
`cont`	NULL (default) for multiple comparison between treatments; 'one-tailed' or 'two-tailed' for corresponding multiple comparisons treatments versus control; partial matching allowed
`data`	a data.frame including the variables used in the formula
`...`	other parameters to be passed as arguments (not used here)

Details

When the value of a Kruskal-Wallis test is significant, it indicates that at least one of the groups is different from at least one of the others. This test helps determining which groups are different with pairwise comparisons adjusted appropriately for multiple comparisons. Those pairs of groups which have observed differences higher than a critical value are considered statistically different at a given significance level. Three types of multiple comparisons are implemented: comparisons between treatments, 'one-tailed' and 'two-tailed' comparison treatments versus control. The first factor level is considered the control. NAs are omitted from data before processing.

For further details please consider the refence below where the method is fully described, or visit `http://pagesperso-orange.fr/giraudoux/#pgirmess` where a copy of the corresponding book section is downloadable.

Value

A list of class 'mc' with the following items:

`statistic`	statistics used
`signif.level`	the significance level
`dif.com`	a data.frame with observed and critical differences

Note

Two alternative methods are proposed in the section 'see also', on François Gillet's suggestion. The three methods do not give necessarily the same results, and the why is still to investigate

References

Siegel and Castellan (1988) Non parametric statistics for the behavioural sciences. MacGraw Hill Int., New York. pp 213-214

See Also

kruskal.test; to reorder factor levels see relevel; for other functions about median multiple comparison see posthoc.kruskal.conover.test, kruskal

Examples

```
resp<-c(0.44,0.44,0.54,0.32,0.21,0.28,0.7,0.77,0.48,0.64,0.71,0.75,0.8,0.76,0.34,0.80,0.73,0.8)
categ<-as.factor(rep(c("A","B","C"),times=1,each=6))
kruskalmc(resp, categ)
kruskalmc(resp, categ, probs=0.01)
kruskalmc(resp, categ, cont="one-tailed")
kruskalmc(resp, categ, cont="two-tailed")

kruskalmc(resp~categ)
kruskalmc(resp~categ, probs=0.01)
kruskalmc(resp~categ, cont="one-tailed")
kruskalmc(resp~categ, cont="two-tailed")
```

---

ks.gof *Kolmogorof-Smirnov goodness of fit test to normal distribution*

Description

Kolmogorof-Smirnov goodness of fit test to normal distribution

Usage

```
ks.gof(var)
```

Arguments

var	a numeric vector

Details

A wrapper of ks.test()

Value

A list with class '"htest"' containing the following components:

`statistic`	the value of the test statistic.
`p.value`	a character string indicating what type of test was performed.
`alternative`	a character string describing the alternative hypothesis.
`method`	a character string indicating what type of test was performed.
`data.name`	a character string giving the name(s) of the data.

References

see ks.test

See Also

`ks.test`

Examples

```
x<-rnorm(50)
ks.gof(x)
```

`mergeTrackObs` *Merge two SpatialPoints or SpatialPointsDataFrame objects, one modelling a track, the other observations*

Description

Merge two SpatialPoints or SpatialPointsDataFrame objects, one modelling a track, the other observations.

Usage

```
mergeTrackObs(sppdfInt,sppdfObs,obscol=NULL)
```

Arguments

`sppdfInt`	A `SpatialPoints` object (the track)
`sppdfObs`	A `SpatialPoints` or `SpatialPointsDataFrame` object (the observations)
`obscol`	The column number in which the number of observations at this point can be found in sppdfObs

Details

Road site counts or faeces collections are often carried out along tracks (paths, roads, transects, etc.). Tracks can be discretized in regular intervals e.g. with `transLines2pix` or `thintrack`, each point being an interval centre. mergeTrackObs uses such a discretized track and sums observations to their nearest track interval. The output is a SpatialPointsDataFrame where each point corresponds to the centre of one track interval. The number of observations in each interval is given in the attribute file. If the number of observations at an observation point can be 0 or any positive number, use obscol to identify the column of sppdfObs where this number is stored.

Value

A SpatialPointsDataframe, with the following attributes:

- ID ID number
- nObs The number of observations in the interval

See Also

`transLines2pix`, `thintrack`

Examples

```
    # track
    library(sp)
     l2 = cbind(c(1,2,3),c(1,1.5,1))
     Sl2 = Line(l2)
     S2 = Lines(list(Sl2), ID="b")
     Sl = SpatialLines(list(S2))
     plot(Sl, col = "blue")
    #observations
     obs <- structure(list(ID = 1:15, long = c(1.04609377280342, 1.0890625305741,
1.08125002916125, 1.24921880953755, 1.34687507719818, 1.50312510545521,
1.88984392539134, 2.37812526369453, 2.39375026652023, 2.36640651157525,
2.38593776510738, 2.62031280749291, 2.69843782162142, 2.85078159917202,
2.90546910906198), lat = c(1.04062476682306, 1.05624976964876,
1.03671851611663, 1.13828103448369, 1.16562478942867, 1.26718730779574,
1.43124983746561, 1.32968731909855, 1.32187481768569, 1.30624981485999,
1.28281231062144, 1.20468729649293, 1.13828103448369, 1.08749977530016,
1.03671851611663)), .Names = c("ID", "long", "lat"), row.names = c(NA,
-15L), class = "data.frame")
    points(obs[,2:3],col="red")
    coordinates(obs)<-~long+lat
obs@data$n<-c(3,4,0,1,1,5,6,4,3,4,4,7,2,2,1) # possibly a count on each location


# examples

# one observation on each location
track<-transLines2pix(Sl,0.1)
trackObs<-mergeTrackObs(track,obs)
```

```
par(mfrow=c(1,2))
plot(Sl)
plot(track,add=TRUE,col="blue")
plot(obs,add=TRUE,col="red",pch=1)

plot(Sl)
plot(track,add=TRUE,col="blue")
plot(trackObs,cex=trackObs@data$nObs,pch=19, col="red",add=TRUE)


# 0 or more observations on each location
obs@data$n<-c(3,4,0,1,1,5,6,4,3,4,4,7,2,2,1) # possibly a count on each location
obs@data
trackObs<-mergeTrackObs(track,obs,obscol=2)

par(mfrow=c(1,2))
plot(Sl)
plot(track,add=TRUE,col="blue")
plot(obs,add=TRUE,col="red",pch=1)

plot(Sl)
plot(track,add=TRUE,col="blue")
plot(trackObs,cex=trackObs@data$nObs/3,pch=19, col="red",add=TRUE)
```

pairsrp *Produces a matrix of scatterplot, regression coefficient and p(Ho)*

Description

Produces a matrix with scatterplot, regression line and a loess smooth in the upper right panel; correlation coefficient (Pearson, Spearman or Kendall) and the probability of Ho in the lower left panel

Usage

```
pairsrp(dataframe, meth = "spearman", pansmo = FALSE, abv = FALSE, lwt.cex = NULL, ...)
```

Arguments

`dataframe`	a data.frame of numeric values
`meth`	a character string indicating which correlation coefficient is to be computed. One of 'pearson', 'kendall', or 'spearman'(default). Can be abbreviated.
`pansmo`	True if a loess smooth is to be plotted. Default to False.
`abv`	True if the variable names must be abbreviates. Default to False.
`lwt.cex`	character size expansion in the lower panel.
`...`	graphical parameters can be given as arguments to 'plot'.

Details

This function is a wrapper for pairs() and cor()

See Also

`pairs`

Examples

```
data(iris)
pairsrp(iris[,1:4],meth="pears",pansmo=TRUE,abv=TRUE)
```

`pave` *Provide square polygons or their node coordinates along a segment*

Description

Provide a user-defined cell grid of polygon squares (or square node points) along a segment. This can be used to define a sampling grid for spatial analysis.

Usage

```
pave(cordseg, yc, xc, fix.edge=NULL, ydown = TRUE, output = "list")
```

Arguments

`cordseg`	the segment coordinates. This can be a vector c(x1,y1,x2,y2), a 2 x 2 matrix or a data.frame (each line a coordinate)
`yc`	the number of segment divisions (y cells)
`xc`	the number of columns (x cells)
`fix.edge`	the edge length of a cell (user specified, default to NULL)
`ydown`	if TRUE (default) squares are computed decreasing y
`output`	a character string indicating which output is required. One of "list", "points" or "spdf". Partial match allowed

Details

The segment must have x1 < x2. If not, it is automatically reordered. When "spdf" is selected the output is an object of class `SpatialPolygonsDataFrame`. The value of the edge length of a cell can passed with the argument fix.edge. In this case, the coordinates of the segment right top are re-computed to adjust the cell edge to an user defined fixed value.

Value

According to the output selected, a list of polygon coordinates, a 2 column matrix with the nodes coordinates or a `SpatialPolygonsDataFrame`.

See Also

SpatialPolygonsDataFrame-class, readShapePoly, readOGR,over, diag2edge

Examples

```
# segment sloping up
coord<-matrix(c(20,20,90,90),nr=2,byrow=TRUE)
plot(coord,type="n",xlim=c(0,100),ylim=c(0,110),asp=1)
lines(coord)
# point grids
gr<-pave(coord,20,4,output="points") # y decreasing
points(gr)
gr<-pave(coord,20,4,output="points",ydown=FALSE) # y increasing
points(gr,col="blue")
# square polygon grids
gr<-pave(coord,20,4) # y decreasing
for (i in 1:length(gr)) polygon(gr[[i]])
gr<-pave(coord,20,4,ydown=FALSE) # y increasing
for (i in 1:length(gr)) polygon(gr[[i]],border="blue")

# segment sloping down
coord<-matrix(c(20,90,90,20),nr=2,byrow=TRUE)
plot(coord,type="n",xlim=c(0,100),ylim=c(0,110),asp=1)
lines(coord)

# point grids
gr<-pave(coord,20,4,output="points")  # y decreasing
points(gr)
gr<-pave(coord,20,4,output="points",ydown=FALSE) # y increasing
points(gr,col="blue")

# fixed edge
plot(coord,type="n",xlim=c(0,100),ylim=c(0,110),asp=1)
lines(coord)
gr<-pave(coord,20,4,fix.edge=4,output="points")
points(gr,col="blue")

plot(coord,type="n",xlim=c(0,100),ylim=c(0,110),asp=1)
lines(coord)
gr<-pave(coord,20,4,fix.edge=5.5,output="points")
points(gr,col="red")

# square polygon grids
coord<-matrix(c(20,90,90,20),nr=2,byrow=TRUE)
plot(coord,type="n",xlim=c(0,100),ylim=c(0,110),asp=1)
lines(coord,lwd=2)
gr<-pave(coord,20,4)# y decreasing
for (i in 1:length(gr)) polygon(gr[[i]])
gr<-pave(coord,20,4,ydown=FALSE) # y increasing
for (i in 1:length(gr)) polygon(gr[[i]],border="blue")
```

```
## Not run:
# Writing a polygon shapefile
gr<-pave(coord,20,4,output="spdf") # y decreasing
library(maptools)
writePolyShape(gr, "myshapefilename")

## End(Not run)
```

pclig — *Compute the percentage of each cell of a matrix or data.frame by row*

Description

Compute the percentage of each cells of a matrix or data.frame by row

Usage

```
pclig(matr)
```

Arguments

`matr`	a matrix or a data.frame

Details

Compute the percentage of each cells of a matrix by row. NA are removed.

Value

Return a matrix with percentages in each cell

See Also

`prop.table`

Examples

```
x<-c(2,10,7,8,7)
y<-c(56,22,7,20,5)
pclig(cbind(x,y))
```

permcont *Random permutation of a contingency table n row x 2 columns*

Description

Return a random permutation of a contingency table n rows x 2 columns keeping the marginal totals

Usage

```
permcont(Table)
```

Arguments

`Table`	a contingency table

Details

The contingency table is split in a two colums table of 0/1 categories, sampled and re-organised with the function table()

Value

A matrix with the permuted values

Examples

```
tab<-cbind(n1=c(10,12,8,7,5),n2=c(4,5,8,10,12))
tab
permcont(tab)
```

PermTest *Permutation test for lm, lme and glm (binomial and Poisson) objects*

Description

Permutation test for lm, lme and glm (binomial and Poisson) objects

Usage

```
    PermTest(obj, B=1000,...)

    ## S3 method for class 'lm'
PermTest(obj, B=1000,...)
    ## S3 method for class 'lme'
PermTest(obj, B=1000,...)
    ## S3 method for class 'glm'
PermTest(obj, B=1000,...)
```

Arguments

`obj`	an object of class lm, lme, or glm
`B`	number of permutations, default = 1000
`...`	used to pass other arguments

Details

For glm, when the response is a two-column matrix with the columns giving the numbers of successes and failures, PermTest.glm uses permcont(); PermTest.lme requires the library nlme.

Value

A list object of class PermTest including:

`p.value`	the p value obtained
`B`	the number of permutations
`call`	the call

Warning

This generic function is implemented in R language, thus can be quite slow.

Note

The implementation of PermTest.lme has been helped by Renaud Lancelot

Examples

```
library(MASS)
mylm<-lm(Postwt~Prewt,data=anorexia)
PermTest(mylm,B=250)


## Dobson (1990) Page 93: Randomized Controlled Trial :
    counts <- c(18,17,15,20,10,20,25,13,12)
    outcome <- gl(3,1,9)
    treatment <- gl(3,3)
    glm.D93 <- glm(counts ~ outcome + treatment, family=poisson)
    PermTest(glm.D93,B=100)

library(nlme)
fm2 <- lme(distance ~ age + Sex, data = Orthodont, random = ~ 1)
PermTest(fm2,B=100)
```

`piankabio` *Computes the Pianka's index of niche overlap*

Description

Computes the Pianka's index of niche overlap

Usage

```
piankabio(dataframe1, dataframe2)
```

Arguments

`dataframe1`	a data frame of two columns: column 1 = dietary category, column 2 = biomass
`dataframe2`	a data frame of two columns: column 1 = dietary category, column 2 = biomass

Details

Computes the Pianka's index of niche overlap

Value

Return the Pianka's index

References

Pianka R.D. 1973 The scructure of lizard communities. Annual Review of Ecology and Systematics, 4: 53-74.

Amroun M., Giraudoux P., Delattre P. 2006 Comparative study of the diets of two sympatric carnivores - the Jackal (Canis aureus) and the Genet (Genetta genetta) - at two sites in Kabylia, Algeria. Mammalia, 70 (3): 247-254

See Also

`piankabioboot`

Examples

```

data(preybiom)
attach(preybiom)
jackal<-preybiom[site=="Y" & sp=="C",5:6]
genet<-preybiom[site=="Y" & sp=="G",5:6]

piankabio(jackal,genet)
```

piankabioboot *Bootstrap Pianka's index*

Description

Bootstrap Pianka's index and return the limits of the empirical confidence interval specified with probs

Usage

```
piankabioboot(dataframe1, dataframe2, B = 1000, probs = c(0.025, 0.975))
```

Arguments

`dataframe1`	a data frame of two columns: column 1 = dietary category, column 2 = biomass
`dataframe2`	a data frame of two columns: column 1 = dietary category, column 2 = biomass
`B`	number of permutations
`probs`	the limits of the confidence interval

Details

Bootstrap Pianka's index and return the limits of the empirical confidence interval sepcified with probs

Value

a vector of the two CI limits

See Also

`piankabio`

Examples

```
data(preybiom)
attach(preybiom)
jackal<-preybiom[site=="Y" & sp=="C",5:6]
genet<-preybiom[site=="Y" & sp=="G",5:6]

piankabioboot(jackal,genet,B=100)
```

polycirc — *Computes the polygon coordinates of a circle*

Description

Computes the polygon coordinates of a circle

Usage

```
polycirc(radius, pts = c(0, 0), nbr = 50)
```

Arguments

`radius`	the length of the radius.
`pts`	the coordinates of the center.
`nbr`	the number of segments required to draw the perimeter

Details

The matrix of coordinates can then be used with the function polygon

Value

A matrix of coordinates.

See Also

`polygon`,`findR`

Examples

```
plot(1:10,1:10,type="n",asp=1)
polygon(polycirc(5),col="blue")
polygon(polycirc(2,c(5,5)), col="red")
```

polycirc2 *Computes the polygon coordinates of a circle sector*

Description

Computes the polygon coordinates of a circle sector

Usage

```
polycirc2(radius = 1, center = c(0, 0), edges = 50, init = pi/2, angle = pi/2)
```

Arguments

`radius`	the circle radius
`center`	the centre coordinates (defaut to x=0, y=0)
`edges`	the circular outline of the sector is approximated by a polygon with this many edges
`init`	number (in radian) specifying the starting angle
`angle`	number (in radian) specifying the sector angle

Details

The matrix of coordinates obtained is intended to be passed to the function `polygon`

Value

A matrix of coordinates

See Also

`polygon`,`polycirc`, `floating.pie`

Examples

```
plot(c(-1,+1),c(-1,+1),type="n",asp=1)
polygon(polycirc2(),col="red")
polygon(polycirc2(init=pi,angle=pi/4),col="green")
polygon(polycirc2(init=1.5*pi,angle=pi/4),col="violet")
polygon(polycirc2(radius=0.5,center=c(0.5,1)),col="blue")

polycirc2(init=pi,angle=pi/4)
```

postxt — *Computes coordinates defined from their relative position on x and y in the plotting region*

Description

Computes coordinates defined from their relative position on x and y in the plotting region

Usage

```
postxt(cd = "ul")
```

Arguments

`cd`	a numerical vector of length 2, values comprised between 0 and 1, or one predefined among "ul", "bl", "ur", "br", "uc", "bc", "ml", "mc", "mr"

Details

The argument cd gives the relative position to be computed in ratio of the x or y axis. For instance c(0.025,0.985) means 2.5 percents on the maximum range of the plot region on x, and 98.5 percents on y (means: close to the upper left corner of the plotting region). Predefined positions are available: "ul", upper left,"bl" bottom left, "ur" upper right, "br" bottom right", "uc" upper center, "bc" bottom center", "ml" medium left, "mc" medium center, "mr" medium right

Value

A list:

`x`	coordinate on x
`y`	coordinate on y

See Also

`text`

Examples

```
plot(rnorm(30),rnorm(30),type="n")
text(postxt("ul"),"here",pos=4)
text(postxt("ur"),"here again",pos=2)
text(postxt("bc"),"again and again")
```

`preybiom` *Jackal and Genet diet in Algeria*

Description

This data set gives the results of dietary analysis performed by Amroun Mansour in two sites of Kabylie, Algeria

Usage

```
data(preybiom)
```

Format

A data frame with 2196 observations on the following variables.

`faeces` a factor for faeces corresponding to faeces identification numbers

`site` a factor for study sites with levels `S` Sebaou `Y` Yacouren

`saison` a factor for seasons with levels `H` `HD` `HP` `S` `SD` `SP`

`sp` a factor for species with levels `C` Jackal `G` Genet

`category` a factor for dietary items with levels `dech` `ind` `ins` `mam` `mol` `oisauv` `oisdom` `rept` `vege` `vegn`

`biomasse` a numeric vector for the weight of each dietary item

References

M. Amroun, P. Giraudoux and P. Delattre 2006 Comparative study of the diets of two sympatric carnivores - the Jackal (Canis aureus) and the Genet (Genetta genetta) - at two sites in Kabylia, Algeria. Mammalia, 70 (3/4): 247-254.

`print.mc` *print method for objects of class 'mc'*

Description

print method for objects of class 'mc'

Usage

```
    ## S3 method for class 'mc'
print(x, ...)
```

Arguments

`x`	an object of class 'mc'
`...`	further arguments to be passed to or from other methods. They are ignored in this function

See Also

`kruskalmc`, `friedmanmc`

Examples

```
resp<-c(0.44,0.44,0.54,0.32,0.21,0.28,0.7,0.77,0.48,0.64,0.71,0.75,0.8,0.76,0.34,0.80,0.73,0.8)
categ<-as.factor(rep(c("A","B","C"),times=1,each=6))
kruskalmc(resp, categ)
```

`QGIS2sp` *Changes a copied-to-clipboard QGIS attribute table into a sp Spatial object or a data.frame*

Description

Reads from the clipboard a copy of the QGIS attribute table of a spatial object (points, linestrings, polygons) and convert it into a sp `Spatial-class` or a `data.frame`.

Usage

```
QGIS2sp(df=FALSE)
```

Arguments

`df`	If TRUE a data.frame is generated with the coordinates of each shape centroid

Value

A `SpatialPointsDataFrame`, `SpatialLinesDataFrame` or `SpatialPolygonsDataFrame` (df=FALSE) or a `data.frame` (df=TRUE) with the two first columns corresponding to the centroid coordinates.

Note

Here, `Spatial-class` are generated without projection attributes (Coordinate Reference System). CRS, if requested, should be added 'manually' (see `proj4string` and `CRS`).

See Also

`readWKT`, `read.delim` with the argument "clipboard", `proj4string`, `CRS`

Examples

```
 if((.Platform$OS.type == "windows") & (interactive())) {
db <-c("wkt_geom\tname", "POINT(104.55 34.60)\tDENG_LING", "POINT(104.45 34.49)\tDIAO_GOU")
 writeLines(db, "clipboard")
QGIS2sp() # to write in the console by hand (if copied and paste, one overwrites the clipboard)
 }
```

---

readGDALbbox *Read a raster using rgdal within a user specified bounding box*

Description

Read a raster using rgdal within a user specified bounding box

Usage

```
readGDALbbox(gdal, spo, mar,...)
```

Arguments

gdal	any raster that can be read by readGDAL
spo	spatial object whose bounding box can be retrieved using bbox
mar	user defined margin around the bounding box (default = 2 pixels)
...	further parameters to pass to readGDAL

Details

This function read a raster file using GDAL within the bounding box of a spatial objet. This permits to extract required subset areas from very large raster data sets that cannot be loaded into the workspace.

Value

returns the required data subset from the raster file as a Spatial object

See Also

readGDAL, bbox

readVista	*Download waypoints and tracks from a GPS*

Description

Download GPS waypoints and tracks using gpsbabel

Usage

```
readVista(i = "garmin", f = "usb:", type="w", SPDF=NULL, invisible=TRUE)
```

Arguments

`i`	INTYPE: a supported file type, default "garmin"
`f`	INFILE: the appropriate device interface, default "usb:"
`type`	"w" waypoints, or "t" track
`SPDF`	if not NULL (the default), characters: the path and filename where to download data as gpx file
`invisible`	Under Windows, do not open an extra window

Details

The function calls gpsbabel via the system. The gpsbabel program must be present and on the user's PATH for the function to work see `http://www.gpsbabel.org`. The function has been tested on the following Garmin GPS devices: Etrex Summit, Etrex Vista Cx and GPSmap 60CSx. On Ubuntu Linux, USB-to-RS232 converter cables were connected succesfully with "/dev/ttyUSB0"; on Windows commonly "com4:" or similar.

Value

If SPDF = NULL (the default) a data frame of four columns:

`ident`	waypoint names or track IDs
`long`	longitude
`lat`	latitude
`altitude`	elevation

Information about the data type (waypoints or tracks) and the date of download are stored as attributes.

If a path and filename is specified with the argument SPDF (e.g. SPDF="./mydata.gpx"), GPS data are downloaded as gpx file.

References

`http://www.gpsbabel.org`

See Also

readGPS

Examples

```
## Not run:
   # a GPS device must be connected
mywaypoints<-readVista() # download waypoints
mytracks<-readVista(type="t") # download tracks

## End(Not run)
```

---

rmls — *Select objects in the parent frame and remove them.*

Description

Select objects in the parent frame and remove them.

Usage

```
rmls()
```

Details

This function has no arguments. This brings up a modal dialog box with a (scrollable) list of objects available in the parent frame. They can be selected by the mouse and then removed.

See Also

ls, rm

Examples

```
toremove<-NULL
ls()
if(interactive()) rmls() # select the object 'toremove' and click OK
ls()
```

rwhatbufCat

Extract the contents of a SpatialPixelsDataFrame or a SpatialGridDataFrame of categorical values within various buffer sizes centred on spatial points

Description

Extract the contents of a SpatialPixelsDataFrame or a SpatialGridDataFrame of categorical values within various buffer sizes centred on points

Usage

```
rwhatbufCat(rast, sites, bufsizes, att=1)
```

Arguments

`rast`	object of class `SpatialPixelsDataFrame` or `SpatialGridDataFrame` to analyse
`sites`	object of class inheriting from `SpatialPoints` containing the points on which buffers must be centred
`bufsizes`	a vector of buffer radii, e.g. c(500, 1000, 1500)
`att`	column number of the attribute variable

Details

This function generates a data.frame with the frequency of each category of a raster map within various radius buffers centred on spatial points.

Value

A dataframe, with the buffer size as first column, the site ID as second column. The other colums are the pixel frequency of each category

See Also

`over`, `rwhatbufNum`, `rwhatbufCat2`

Examples

```
# raster creation
library(sp)
data(meuse.grid)
coordinates(meuse.grid) = ~x+y
gridded(meuse.grid) = TRUE

# random selection of points within the raster area
mypoints<-spsample(meuse.grid,n=10,type="random") # random points are appx 10, see spsample doc
```

```
mypoints<-SpatialPointsDataFrame(coordinates(mypoints),data.frame(id=1:nrow(mypoints@coords)))

image(meuse.grid,att=4,col=c("red","green","blue")) # soil map
plot(mypoints,add=TRUE) # points
  # get the number of pixels of each category in each buffer
rwhatbufCat(meuse.grid,mypoints,c(500,1000),att=4)
```

---

rwhatbufCat2 — *Extract the contents of a raster file of categorical values within various buffer sizes centred on spatial points*

Description

Extract the contents of a raster file of categorical values within various buffer sizes centred on spatial points

Usage

```
rwhatbufCat2(rast, sites, bufsizes, att=1, asList=FALSE)
```

Arguments

`rast`	name of the raster file readable with rgdal to analyse
`sites`	object of class inheriting from `SpatialPoints` containing spatial points on which buffers must be centered
`bufsizes`	a vector of buffer radii, e.g. c(500, 1000, 1500)
`att`	column number of the attribute variable
`asList`	if TRUE the output is a list else a data.frame (default)

Details

This function does not load the full raster file into the memory but loads sequentially subsets corresponding to the size of each buffer. This allows proceeding massive rasters that cannot be loaded into RAM in full. It generates either a data.frame or a list of lists giving for each buffer size (top level of the list of lists) the number of pixels of each category value within the buffer at each point site.

The function reads the raster file on the hard disk as many times as buffers to compute. Thus, computation time is about 5 times longer than `rwhatbufCat`. Empty buffer (no pixel inside) gives (so far) anavoidable topology and dimension errors at reading and stop computation. User must ensure size and location so that every buffer includes at least one pixel.

Value

A data.frame or a list of lists giving for each buffer size (top level of the list of lists) pixel frequency for each category within the buffer at each spatial point

See Also

over, rwhatbufNum, rwhatbufCat

Examples

```
library(sp)

myrastername<-system.file("pictures/SP27GTIF.TIF", package = "rgdal")[1]

mylocations<-structure(list(x = c(694728, 684662, 702339, 691819, 700091),
y = c(1906654, 1886491, 1884426, 1884373, 1886872)),
.Names = c("x", "y"), row.names = c(NA, -5L), class = "data.frame")

coordinates(mylocations)<-~x+y

result<-rwhatbufCat2(myrastername,mylocations,c(500,1000))
result

result<-rwhatbufCat2(myrastername,mylocations,c(500,1000),asList=TRUE)
result[[1]] # results for buffer 500 (5 buffer centers)
result[[2]] # results for buffer 1000 (5 buffer centers)
```

---

rwhatbufNum — *Extract the contents of a SpatialPixelsDataFrame or a SpatialGridDataFrame of numerical values within various buffer sizes centred on spatial points*

Description

Extract the contents of a SpatialPixelsDataFrame or a SpatialGridDataFrame of numerical values within various buffer sizes centred on spatial points

Usage

```
rwhatbufNum(rast, sites, bufsizes, att=1)
```

Arguments

rast	object of class SpatialPixelsDataFrame or SpatialGridDataFrame to analyse
sites	object of class SpatialPointsDataFrame containing spatial points on which buffers must be centred
bufsizes	a vector of buffer radii, e.g. c(500, 1000, 1500)
att	column number of the attribute variable

Details

This function generates a list of lists giving for each buffer size (top level in the list of lists) the values of the raster map for each point site within the buffer.

Value

A list of lists: top level, the buffer size; second level, the values of the raster map for each spatial point site within the buffer

See Also

`over`, `rwhatbufCat`, `rwhatbufCat2`

Examples

```
library(pgirmess)
# raster creation
library(sp)
data(meuse.grid)
coordinates(meuse.grid) = ~x+y
gridded(meuse.grid) = TRUE

# random selection of points within the raster area
mypoints<-spsample(meuse.grid,n=10,type="random") # random points are appx 10, see spsample doc
mypoints<-SpatialPointsDataFrame(coordinates(mypoints),data.frame(id=1:nrow(mypoints@coords)))

image(meuse.grid,att=3) # distance to the river
plot(mypoints,add=TRUE,pch=20,cex=0.1) # points
for (i in 1:nrow(mypoints@coords)){
polygon(polycirc(50, mypoints@coords[i,]),border="blue") # buffer 50 place
}
for (i in 1:nrow(mypoints@coords)) {
polygon(polycirc(100, mypoints@coords[i,]),border="green") # buffer 100 place
}
mybuffers<-rwhatbufNum(meuse.grid,mypoints,c(50,100),att=3) # get the values in each buffer

names(mybuffers) # two list given

mybuffers[[1]] # list of values for each point (buffer 50)
mybuffers[[1]][[1]] # list of values for the first buffer 50 (point #1)
```

`Segments` *Draw line segments between pairs of points.*

Description

Draw line segments between pairs of points from a vector, matrix or data frame of 4 points coordinates x0, y0, x1, y1

Usage

```
Segments(mydata, ...)
```

Arguments

mydata	a vector, matrix or data frame
...	further graphical parameters (from 'par')

Details

a wrapper to 'segments' to handle coordinates passed as vector, matrix or data frame. Any vector is turned into a matrix of four columns.

See Also

segments

Examples

```
mydata<-cbind(rnorm(20),rnorm(20),rnorm(20),rnorm(20))
plot(range(rbind(mydata[,1],mydata[,3])),range(rbind(mydata[,2],mydata[,4])),
type="n",xlab="",ylab="")
Segments(mydata,col=rainbow(20))

myvec<-rnorm(4)
plot(myvec[c(1,3)],myvec[c(2,4)],type="n",xlab="",ylab="")
Segments(myvec)

myvec<-rnorm(16)
plot(myvec,myvec,type="n",xlab="",ylab="")
Segments(myvec)
```

---

selMod *Model selection according to information theoretic methods*

Description

Handles lm, glm and list of e.g. lm, glm, nls, lme and nlme objects and provides parameters to compare models according to Anderson et al. (2001)

Usage

```
    selMod(aModel, Order = "AICc", ...)

    ## S3 method for class 'lm'
selMod(aModel, Order = "AICc", dropNull = FALSE, selconv=TRUE, ...)
    ## S3 method for class 'list'
selMod(aModel, Order = "AICc", ...)
```

Arguments

`aModel`	a lm or glm model or a list of relevant models (see details)
`dropNull`	if TRUE, drops the simplest model (e.g. y 1)
`Order`	if set to "AICc" (default) sort the models on this parameter, otherwise "AIC" is allowed
`selconv`	if TRUE (default) keep the models for which convergence is obtained (glm object only) and with no anova singularity (lm and glm)
`...`	other parameters to be passed as arguments (not used here)

Details

This function provides parameters used in the information theoretic methods for model comparisons.

- .lm and glm objects can be passed directly as the upper scope of term addition (all terms added). Every model from y∼1 is computed adding one term at a time until the upper scope model is derived. This is a stepwise analysis where the terms are added sequentially and this does NOT provide all combinations of terms and interactions. Offset terms cannot be proceeded here.
- .A list of user specified lm, glm, nls, lme or nlme objects (actually any object for which AIC and logLik functions are applicable) to compare can alternately be passed.

Value

A dataframe including:

`LL`	the maximized log-likelihood
`K`	the number of estimated parameters
`N2K`	the number of observations/K
`AIC`	the Akaike index criterion
`deltAIC`	the difference between AIC and the lowest AIC value
`w_i`	the Akaike weights
`deltAICc`	the difference between AICc and the lowest AICc value; advised to be used when n2K < 40
`w_ic`	the AICc weights

The models examined from first to last are stored as attribute

Author(s)

Patrick Giraudoux and David Pleydell: pgiraudo@univ-fcomte.fr, david.pleydell@inra.fr

References

- .Anderson, D.R., Link, W.A., Johnson, D.H. and Burnham, K.P. (2001). Suggestions for presenting the results of data analyses. Journal of Wildlife Management, 65, 373-378
- .Burnham, K.P. and Anderson, D.R. (2002) Model Selection and Multimodel Inference: a Practical Information-Theoretic Approach, 2nd edn., Springer-Verlag, New York. 353 pp

See Also

AIC,logLik, aictab

Examples

```
library(MASS)
anorex.1 <- lm(Postwt ~ Prewt*Treat, data = anorexia)
selMod(anorex.1)
anorex.2 <- glm(Postwt ~ Prewt*Treat, family=gaussian,data = anorexia)
selMod(anorex.2)
anorex.3<-lm(Postwt ~ Prewt+Treat, data = anorexia)
mycomp<-selMod(list(anorex.1,anorex.2,anorex.3))
mycomp
attributes(mycomp)$models
```

---

shannon *Computes Shannon's and equitability indices*

Description

Computes Shannon's and equitability indices

Usage

```
shannon(vect, base=2)
```

Arguments

vect	a probability vector whose sum = 1 or a frequency vector
base	logarithm base used (default=2)

Details

Computes Shannon's and equitability indices. The vector passed can be a probability vector whose sum equal 1 or a vector of frequencies (e.g. the number of food item of each category).

Value

A vector of two values: Shannon's and equitability indices. The base logarithm used is stored as attribute

See Also

`shannonbio`

Examples

```
x<-c(0.1,0.5,0.2,0.1,0.1)
sum(x)
shannon(x)

x<-rpois(10,6)
shannon(x, base=exp(1))
```

`shannonbio` *Computes Shannon's and equitability indices from a data frame of dietary analysis (n, biomass,...)*

Description

Computes Shannon's and equitability indices from a data frame of two columns: column 1, dietary category; column 2, abundance (n, biomass,...)

Usage

```
shannonbio(data1)
```

Arguments

`data1`	a data frame of two columns: column 1, dietary category; column 2, abundance (n, biomass,...)

Details

Computes Shannon's and equitability indices from a data frame of two columns: column 1, dietary category; column 2, abundance (n, biomass,...)

Value

A vector of two values: Shannon's and equitability indices

See Also

`shannon`, `difshannonbio`

Examples

```
data(preybiom)
shannonbio(preybiom[,5:6])
```

---

shannonbioboot *Boostrap Shannon's and equitability indices*

Description

Boostrap Shannon's and equitability indices and return an object of class boot. Confidence intervals can be computed with boot.ci().

Usage

```
shannonbioboot(data1, B = 1000)
```

Arguments

`data1`	a data frame of two columns: column 1, dietary category; column 2, abundance (n, biomass,...)
`B`	number of permutations

Details

Boostrap Shannon\'s and equitability indices and return an object of class boot. Confidence intervals can be computed with boot.ci(). Requires the boot library.

Value

An object of class boot including the bootstrap statistics for H' (t1*) and J' (t2*)

See Also

`boot`, `boot.ci`, `shannonbio`

Examples

```
data(preybiom)
myboot<-shannonbioboot(preybiom[,5:6],B=100)
library(boot)
boot.ci(myboot, index=1,type=c("norm","basic","perc")) # confidence intervals for H'
boot.ci(myboot, index=2,type=c("norm","basic","perc")) # confidence intervals for J'
```

`siegelp179` *Data on rats training*

Description

Ranks of 18 matched groups of rats after training under three methods of reinforcement.

Usage

```
data(siegelp179)
```

Format

A data frame with 54 observations on the following 3 variables.

block Group (each of three litter mates)

treatment A factor for the type of reinforcement with levels `RR` `RU` `UR`

score Speed of transfer to another behaviour (the lower, the better the learning)

Details

18 blocks made of three rats of the same litter, each being given a different learning pattern (RR, RU or UR)

Source

Grosslight J.H. and Radlow R. (1956) Patterning effect of the nonreinforcement-reinforcement sequence in a discrimination situation. Journal of Comparative and Physiological Psychology, 49: 542-546 in Siegel & Castellan 1988. Non parametric statistics for the behavioural sciences. Mc Graw Hill Int. Edt.

Examples

```
data(siegelp179)
```

`tabcont2categ` *Convert a contingency table (data.frame) into a presence/absence table of categories*

Description

Convert a contingency table (data frame) into a data.frame of factors

Usage

```
tabcont2categ(tab)
```

Arguments

`tab`	A data.frame (contingency table)

Details

Convert a contingency table (data frame) into a data.frame of factors

Value

A data frame

Examples

```
mydata<-as.data.frame(matrix(rpois(9,5),nr=3,nc=3))
names(mydata)<-LETTERS[1:3]
row.names(mydata)<-letters[1:3]

tabcont2categ(mydata)
```

`thintrack` *Thin a track just keeping the points separated by a user defined minimal distance*

Description

Thin a track stored as a `SpatialPointsDataFrame` object, just keeping the points separated by a user defined minimal distance.

Usage

```
thintrack(spdf,mindist=100)
```

Arguments

`spdf`	a `SpatialPointsDataFrame` of point tracks
`mindist`	minimal distance requested between two points (default = 100)

Details

Tracks downloaded from GPS often provide an unnecessary large density of points at irregular distances. This function starts reading from the first point of the track and removes all points within a user specified radius (USR), then reads the next point and removes all points within the USR, and so on...

Value

A `SpatialPoints` object of the track thinned.

See Also

mergeTrackObs

Examples

```
library(sp)
mySPDF<-structure(list(x = c(748775, 748807, 748834, 748854, 748871,
748873, 748880, 748910, 748919, 748917, 748921, 748923, 748924,
748921, 748921, 748921, 748922, 748915, 748616, 748613, 748612,
748613, 748613, 748615, 748613, 748616, 748615, 748618, 748615,
748619, 748618, 748620, 748586, 748553, 748494, 748444, 748424,
748366, 748305, 748305), y = c(105716, 105761, 105808, 105856,
105911, 105964, 106019, 106065, 106114, 106167, 106219, 106274,
106329, 106385, 106441, 106494, 106550, 106571, 105835, 105779,
105723, 105665, 105600, 105537, 105473, 105412, 105350, 105293,
105234, 105180, 105123, 105070, 105023, 104960, 104956, 104947,
104906, 104905, 104901, 104904), ID = 1:40), .Names = c("x",
"y", "ID"), row.names = c("1", "2", "3", "4", "5", "6", "7",
"8", "9", "10", "11", "12", "13", "14", "15", "16", "17", "18",
"19", "20", "21", "22", "23", "24", "25", "26", "27", "28", "29",
"30", "31", "32", "33", "34", "35", "36", "37", "38", "39", "40"
), class = "data.frame")
coordinates(mySPDF)<-~x+y

plot(mySPDF,pch=19,cex=0.5)
plot(thintrack(mySPDF),pch=19,cex=0.7,col="red",add=TRUE)

plot(mySPDF,pch=19,cex=0.5)
plot(thintrack(mySPDF,min=200),pch=19,cex=0.7,col="red",add=TRUE)
```

---

trans2pix *Convert a transect coordinate file with some waypoints separated by NA into a matrix with intermediate coordinates replacing NA.*

Description

Convert a transect coordinate file with some waypoints separated by NA into a matrix with intermediate coordinates replacing NA.

Usage

```
trans2pix(vect)
```

Arguments

vect	A two column matrix or data.frame

Details

If vect has more than two column the two first column only are read. This function computes the intermediate coordinates between two waypoints replacing NA values.

Value

A matrix with the intermediate coordinates computed.

See Also

`trans2seg`

Examples

```
x<-c(10,NA, NA, NA,56,NA,NA,100)
y<-c(23,NA, NA, NA,32,NA,NA,150)
cols=c("red","blue","blue","blue","red","blue","blue","red")
plot(x,y,col=cols,pch=19)
plot(trans2pix(cbind(x,y)),col=cols,pch=19)
```

`trans2seg` *Convert a transect coordinate file into a matrix with segment coordinates.*

Description

Convert a transect coordinate file (e.g.: waypoints) into a matrix with segment coordinates.

Usage

```
trans2seg(vect)
```

Arguments

`vect`	A two column matrix or data.frame

Details

The argument passed is a matrix or data.frame of two columns each row is a transect interval; each column must start (first row) and end (last row) with a landmark ; intermediate waypoints must have coordinates in the two columns of the row. Other rows must be NA values.

Value

A matrix of 4 columns to be passed e.g. to functions as "segments".

See Also

trans2pix

Examples

```
x<-c(10,NA, NA, NA,56,NA,NA,100)
y<-c(23,NA, NA, NA,32,NA,NA,150)
cols=c("red","blue","blue","blue","red","blue","blue","red")
plot(x,y,col=cols,pch=19)
mysegs<-trans2seg(cbind(x,y))
segments(mysegs[,1],mysegs[,2],mysegs[,3],mysegs[,4])
```

`transLines2pix` *Convert a SpatialLines or a SpatialLinesDataFrame object into SpatialPointsDataFrame with spatial points at regular distance along the lines*

Description

Convert a SpatialLines or a SpatialLinesDataFrame object into SptaialPointsDataFrame with spatial points at regular distance along the lines

Usage

```
transLines2pix(spldf,mindist=100)
```

Arguments

`spldf`	A `SpatialLines` or a `SpatialLinesDataFrame`
`mindist`	the distance between two points (default to 100)

Details

This function can be used e.g to discretize track lines (roads, paths, transects, etc.) into series of regular points. Each point is the centre of one interval. Beware that if mindist is larger than the distance between nodes, the output file will keep the nodes. Furthermore, mindist might not be fully respected in some spatial configurations. In such cases to filter points using `thintrack` complementarily is advised.

Value

A `SpatialPointsDataFrame`

See Also

`trans2pix`, `thintrack`, `mergeTrackObs`

Examples

```
# from the sp vignette:
     library(sp)
     l1 = cbind(c(1,2,3),c(3,2,2))
     l1a = cbind(l1[,1]+.05,l1[,2]+.05)
     l2 = cbind(c(1,2,3),c(1,1.5,1))
     Sl1 = Line(l1)
     Sl1a = Line(l1a)
     Sl2 = Line(l2)
     S1 = Lines(list(Sl1, Sl1a), ID="a")
     S2 = Lines(list(Sl2), ID="b")
     Sl = SpatialLines(list(S1,S2))
     plot(Sl, col = c("red", "blue"))

     trpt<-transLines2pix(Sl,mindist=0.1)
     plot(trpt,add=TRUE)
```

TukeyHSDs *Simplify the list of a TukeyHSD object keeping the significant differences only.*

Description

Simplify the list of a TukeyHSD object keeping the significant differences only.

Usage

```
TukeyHSDs(TukeyHSD.object)
```

Arguments

`TukeyHSD.object`	An object of calls "TukeyHSD"

Details

When TukeyHSD is used on a fitted model with large numbers of categories, the number of pairwise comparisons is extremely large (n(n-1)/2). TukeyHSDs simplify the TukeyHSD object keeping the significant pairwise comparisons only. A plot method exists for TukeyHSD objects.

Value

An object of class "multicomp" and "TukeyHSD"

See Also

`TukeyHSD`

Examples

```
summary(fm1 <- aov(breaks ~ wool + tension, data = warpbreaks))
myobject<-TukeyHSD(fm1, "tension", ordered = TRUE)
myobject
TukeyHSDs(myobject)
```

---

uploadGPS *Upload waypoints to Garmin GPS*

Description

Upload waypoints to Garmin GPS, using gpsbabel

Usage

```
uploadGPS(gpx, f = "usb:",type="w")
```

Arguments

gpx	name of the .gpx file (can be created from a data frame using writeGPX)
f	the appropriate device interface, default "usb:", see details
type	'w' for waypoints (default), 't' for track

Details

This function uploads waypoints or a track to a garmin GPS from a '.gpx' file. gpsbabel is called via the system. Therefore gpsbabel must be installed and on the user's path, see http://www.gpsbabel.org. If not the default, device interface should be something as "usb:", "usb:1", "com:4" or on linux "/dev/ttyUSB0", etc.

Warning

Overwrite waypoints having the same name in the GPS

See Also

writeGPX

Examples

```
## Not run:
  # a GPS device must be connected
coords<-data.frame(ID=c("C18J01", "C18J02"),Long= c(-46.996602, 47.002745),
Lat=c(-6.148734, 6.14829),Alt=c(250,1230))

writeGPX(coords,"mywaypoints")
uploadGPS("mywaypoint.gpx")

## End(Not run)
```

`val4symb` *Center a numerical vector on a parameter position and provides absolute values and colors according to negative and positive values*

Description

Center a numerical vector on a parameter position and provides absolute values and colors according to negative and positive values

Usage

```
val4symb(x, FUN=mean, col = c("blue", "red"),...)
```

Arguments

`x`	a numerical vector
`FUN`	a function computing a position parameter, typically `mean` or `median`. Default to `mean`
`col`	a character vector of 2 values, default=c("blue","red"), blue for <0, red for >=0
`...`	optional arguments to 'FUN'

Value

A list with

`size`	the absolute values of the difference to the position parameter (eg mean, median)
`col`	a character vector with 2 colors, each corresponding to positive or negative values

See Also

`symbols`, `mean`, `median`, `scale`

Examples

```
x<-rnorm(30)
y<-rnorm(30)

z<-val4symb(rnorm(30))
symbols(x,y,circle=z$size,inches=0.2,bg=z$col)

z<-val4symb(scale(rnorm(30)))
symbols(x,y,circle=z$size,inches=0.2,bg=z$col)

z<-val4symb(rnorm(30),col=c("green","violet"))
symbols(x,y,circle=z$size,inches=0.2,bg=z$col)

z<-val4symb(rnorm(30),trim=0.025)
symbols(x,y,circle=z$size,inches=0.2,bg=z$col)

z<-val4symb(rnorm(30),median)
symbols(x,y,circle=z$size,inches=0.2,bg=z$col)

myfun<-function(x) 20 # passes an arbitrary constant
z<-val4symb(1:30,myfun)
symbols(x,y,circle=z$size,inches=0.2,bg=z$col)
```

---

valchisq *Values of the partial chi-square in each cell of a contingency table*

Description

Computes the values of the partial chi-square in each cell of a contingency table

Usage

```
valchisq(matr)
```

Arguments

matr	a matrix (contingency table)

Details

Computes the values of the chi-square in each cell of a contingency table

Value

A matrix with the chi-square values computed

Note

No correction (e.g. Yate's etc.) is done !

See Also

valat, chisq.test

Examples

```
x <- matrix(c(12, 5, 7, 7), nc = 2)
x
valchisq(x)
```

---

write.delim *Write a data.frame*

Description

Write a simple data.frame into a text file with header, no row.names, fields separated by tab.

Usage

```
write.delim(x, file = "", row.names = FALSE, quote = FALSE, sep = "\t", ...)
```

Arguments

`x`	a data.frame
`file`	a character string for file name
`row.names`	either a logical value indicating whether the row names of 'x' are to be written along with 'x', or a character vector of row names to be written
`quote`	a logical value or a numeric vector. If 'TRUE', any character or factor columns will be surrounded by double quotes. If a numeric vector, its elements are taken as the indices of the columns to quote. In both cases, row and column names are quoted if they are written. If 'FALSE', nothing is quoted.
`sep`	the field separator string. Values within each row of 'x' are separated by this string.
`...`	additional arguments accepted by write.table

Details

Simple wrapper of write.table.

Value

A tab delimited text file.

See Also

`write.table`

Examples

```
data(preybiom)
write.delim(preybiom[1:10,]) # output to the console
write.delim(preybiom[1:10,],file="Myfile.txt") # write a file in the working directory
```

`writecn.delim` *Write a data.frame that has Chinese characters*

Description

Handle a data.frame with Chinese characters and write it into a UTF-8 text file with header, no row.names, tab delimited fields.

Usage

```
writecn.delim(db, file = "",row.names = FALSE, quote = FALSE, sep = "\t", ...)
```

Arguments

`db`	a data.frame
`file`	file name (character string)
`row.names`	either a logical value indicating whether the row names of 'x' are to be written along with 'x', or a character vector of row names to be written
`quote`	a logical value or a numeric vector. If 'TRUE', any character or factor columns will be surrounded by double quotes. If a numeric vector, its elements are taken as the indices of the columns to quote. In both cases, row and column names are quoted if they are written. If 'FALSE', nothing is quoted.
`sep`	the field separator string. Values within each row of 'x' are separated by this string.
`...`	additional arguments to pass to `write.table`

Details

Writing a data.frame into text files can be quite cumbersome if the system locale is not Chinese. This function set up the locale to Chinese, write the data.frame using `write.table` with fileEncoding = "UTF-8", then restore the original locale.

Value

An ascii text file, tab delimited.

See Also

`write.table`, `Sys.setlocale`

`writeGPX` *Convert a data frame into a GPX file of waypoints or track*

Description

Convert a data frame of labels, geographical coordinates and optionally altitude into a GPX file of waypoints or track that can be uploaded to Garmin GPS

Usage

```
writeGPX(x, filename = "",type="w")
```

Arguments

`x`	data.frame of three (optionally four) columns (see details)
`filename`	a character string naming the file to print to. If '""' (the default), prints to the standard output connection, the console (unless redirected by 'sink')
`type`	'w' for waypoints (default) or 't' for track

Details

The data frame must have three (optionally four) columns:

1. character or integer, waypoint ID for waypoints ; column not read for track
2. numeric, longitude (decimal degrees), negative for west
3. numeric, latitude (decimal degrees), negative for south
4. numeric, elevation (meters) (optional)

A suffix '.gpx' is added to the file name if not provided by user. The file obtained can be uploaded to Garmin GPS but cannot be read eg from MapSource for some reasons.

Note

for more standard GPX file, see `writeOGR` with arguments like layer="waypoints", driver="GPX", dataset_options ="GPX_USE_EXTENSIONS=yes" can alternately be used; `readOGR` with arguments like layer="waypoints", drop_unsupported_fields=TRUE

See Also

`writeOGR`

Examples

```
coords<-data.frame(ID=c("C18J01", "C18J02"),Long= c(-46.996602, 47.002745),
Lat=c(-6.148734, 6.14829),Alt=c(250,1230))

writeGPX(coords) # waypoints
writeGPX(coords,type="t") # track
```

`writePRJ` *Write the projection file of a shapefile from a spatial object*

Description

Write the projection file of a shapefile from a spatial object

Usage

```
writePRJ(spobj, filename)
```

Arguments

`spobj`	any spatial object having a CRS extractible with `proj4string`
`filename`	a character string naming the file to print to. If '""' (the default), prints to the standard output connection, the console (unless redirected by 'sink')

Details

A suffix '.prj' is added to the file name if not user provided.

Examples

```
library(sp)
mypoints<-data.frame(long=runif(10,-90,+90),lat=runif(10,-90,+90))
coordinates(mypoints)<-~long+lat # SpatialPoints object
proj4string(mypoints)<-CRS("+proj=longlat +ellps=WGS84 +datum=WGS84") # WGS84 coordinates
writePRJ(mypoints,"")
```

Index