Package ‘pgirmess’

April 30, 2015

Date 2015-04-29
Version 1.6.0
Title Data Analysis in Ecology
Author Patrick Giraudoux <patrick.giraudoux@univ-fcomte.fr>
Maintainer Patrick Giraudoux <patrick.giraudoux@univ-fcomte.fr>
Imports boot (>= 1.3-4), nlme (>= 3.1-64), rgdal (>= 0.7-8), sp (>= 0.9-97), spdep (>= 0.5-43), splancs (>= 2.01-31)
Suggests MASS (>= 7.3-1)
Description Miscellaneous functions for data analysis in ecology, with special emphasis on spatial data.
License GPL (>= 2)
URL http://perso.orange.fr/giraudoux
NeedsCompilation no
Repository CRAN
Date/Publication 2015-04-30 09:00:03

R topics documented:

CI ...................................................... 3
classnum .............................................. 4
clipQGIS2df ........................................... 5
cormat .................................................. 5
correlog .............................................. 6
date2winter ......................................... 8
diag2edge ............................................ 9
difshannonbio ....................................... 10
dirProj ............................................... 11
dirSeg ................................................. 12
distNNeigh .......................................... 13
distNode .............................................. 13
distSeg ............................................... 14
<table>
<thead>
<tr>
<th>R topics documented:</th>
</tr>
</thead>
<tbody>
<tr>
<td>distTot</td>
</tr>
<tr>
<td>expandpoly</td>
</tr>
<tr>
<td>friedmanmc</td>
</tr>
<tr>
<td>gps2gpix</td>
</tr>
<tr>
<td>kruskalmc</td>
</tr>
<tr>
<td>ks.gof</td>
</tr>
<tr>
<td>mergeTrackObs</td>
</tr>
<tr>
<td>pairsrp</td>
</tr>
<tr>
<td>pave</td>
</tr>
<tr>
<td>pclig</td>
</tr>
<tr>
<td>permcont</td>
</tr>
<tr>
<td>PermTest</td>
</tr>
<tr>
<td>piankabio</td>
</tr>
<tr>
<td>piankabioboot</td>
</tr>
<tr>
<td>polycirc</td>
</tr>
<tr>
<td>polycirc2</td>
</tr>
<tr>
<td>postxt</td>
</tr>
<tr>
<td>preybiom</td>
</tr>
<tr>
<td>print.mc</td>
</tr>
<tr>
<td>readGDALbbox</td>
</tr>
<tr>
<td>readVista</td>
</tr>
<tr>
<td>rmls</td>
</tr>
<tr>
<td>rwhatbufCat</td>
</tr>
<tr>
<td>rwhatbufCat2</td>
</tr>
<tr>
<td>rwhatbufNum</td>
</tr>
<tr>
<td>rwhatpoly</td>
</tr>
<tr>
<td>Segments</td>
</tr>
<tr>
<td>selMod</td>
</tr>
<tr>
<td>shannon</td>
</tr>
<tr>
<td>shannonbio</td>
</tr>
<tr>
<td>shannonbioboot</td>
</tr>
<tr>
<td>siegelp179</td>
</tr>
<tr>
<td>tabcont2categ</td>
</tr>
<tr>
<td>thintrack</td>
</tr>
<tr>
<td>trans2pix</td>
</tr>
<tr>
<td>trans2seg</td>
</tr>
<tr>
<td>transLines2pix</td>
</tr>
<tr>
<td>TukeyHSDs</td>
</tr>
<tr>
<td>uploadGPS</td>
</tr>
<tr>
<td>val4symb</td>
</tr>
<tr>
<td>valchisq</td>
</tr>
<tr>
<td>write.delim</td>
</tr>
<tr>
<td>writeGPX</td>
</tr>
<tr>
<td>writePRJ</td>
</tr>
</tbody>
</table>

Index 61
Description

Provides a n x 2 matrix with the lower limit (column 1) and upper limit (column 2) of the 95 percent confidence interval of percentages

Usage

CI(x, y, tottrials = FALSE)

Arguments

- x: a vector with the number of positive observations
- y: a vector of the same length as x with the number of negative observations, or of the total number of observations
- tottrials: if false (the default) y is the number of negative observations; if true, y is the total number of observations

Details

Wrapper of prop.test().

Value

A matrix of length(x) rows and 2 columns. Column 1: lower limit; column 2: upper limit of the 95 percent confidence interval

See Also

prop.test

Examples

```r
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(x, y)

x <- c(2,10,7,8,7) # eg: number of positive cases
y <- c(4,11,7,16,10)# eg: total number of cases
CI(x, y, tottrials=TRUE)
```
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)
clipQGIS2df

Change a copied QGIS attribute table of a point shapefile into a data.frame

Description

Copying and attribute table of a shapefile in QGIS gives a clipboard file that can be read using 'read.delim("clipboard")'. The first column is in wkt.geometry format. If is a POINT geometry function converts this column into two columns giving the longitude and the latitude.

Usage

clipQGIS2df(cb)

Arguments

cb  A data.frame obtained from the copy of an attribute table of a QGIS shapefile

Value

A data.frame with the two first columns corresponding to longitude and latitude, and the following to the shapefile attributes.

Examples

myclip <-
structure(list(wkt_geom = c("POINT(319139 4394340)",
                        "POINT(319144 4394593)")
                        , ID.ligne = c("SM1205SB", "SM1205SC"), altitude = c(3000,
                        3088)), .Names = c("wkt_geom", "ID.ligne", "altitude"), row.names = 1:2, class = "data.frame")

myclip

clipQGIS2df(myclip)

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

Author(s)

Patrick Giraudoux <pgiraudo@univ-fcomte.fr>

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,...)
correlog

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

Author(s)

Patrick Giraudoux <pgiraudo@univ-fcomte.fr>

See Also

`pave`

Examples

```r
# 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
```
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'

Author(s)
patrick.giraudoux <pgiraudo@univ-fcomte.fr>

See Also
shannonbio
**dirProj**

*Computes new coordinates given bearings and distances.*

**Description**

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

**Usage**

```r
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**

Computations 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`

**Examples**

```r
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)
```

**Examples**

```r
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)
```
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 columns 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

```r
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),"\000b0"),cex=0.7)
```
distNNeigh

**Description**

Computes distances to the nearest neighbour

**Usage**

```r
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, knnRnb, nbdists

**Examples**

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

---

distNode

**Description**

Computes the distances between each nodes of a polyline.

**Usage**

```r
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 decdeg is FALSE (default), distance computed is Euclidian. Units depends on the coordinate systems. If decdeg = TRUE, \( D = 1852 \times 60 \times (180/\pi) \times \arccos (\sin(la1) \times \sin(la2) + \cos(la1) \times \cos(la2) \times \cos(\text{abs}(lg1 - lg2)) \). 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

```r
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

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

Arguments

- **mydata**: A matrix or data frame of 4 columns giving the coordinates of each segment top: 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, \( D = 1852 \times 60 \times (180/\pi) \times \arccos (\sin(la1) \times \sin(la2) + \cos(la1) \times \cos(la2) \times \cos(\text{abs}(lg1 - lg2)) \). 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
**distTot**

**Value**

A vector of distances

**See Also**

`distNode, distTot`

**Examples**

```r
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**

```r
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 `decdeg` is FALSE (default), distance computed is Euclidian. Units depends on the coordinate systems. If `decdeg = TRUE`, $D = 1852 \times 60 \times (180/\pi) \times \arccos(\sin(la1) \times \sin(la2) + \cos(la1) \times \cos(la2) \times \cos(abs(lg1 - lg2)))$. 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 (eg 2 = 2 times, 0.5 = half, etc...)

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")
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


See Also

friedman.test
Examples

```r
data(siegelp179)
attach(siegelp179)

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

mymatrix<-matrix(score,nc=3)
friedman.test(mymatrix)
friedmannmc(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

```r
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](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

```r
## Not run:
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

```r
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 obtained 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. Those pairs of groups which have observed differences higher than a critical value are considered statistically different at a given significance level. Three type 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 reference below where the method is fully described. One may also want to visit http://pagesperso-orange.fr/giraudoux/#pgirmess

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

formula method adapted on Derek Ogle’s suggestion

References


See Also

kruskal.test, to reorder factor levels see relevel

Examples

resp<-c(0.44,0.44,0.54,0.32,0.21,0.28,0.7,0.77,0.48,0.64,0.71,0.75,0.8,0.76,0.34,0.8,0.73,0.8)
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")
mergeTrackObs

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.

Usage

mergeTrackObs(sppdfInt, sppdfObs, obscol=NULL)
mergeTrackObs

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

```r
# track
library(sp)
L2 = cbind(c(1,2,3),c(1,1.5,1))
S2 = Line(L2)
S2 = Lines(list(S2), ID="b")
S1 = SpatialLines(list(S2))
plot(S1, col = "blue")
# observations
obs <- structure(list(ID = 1:15, long = c(1.04609377280342, 1.0890625035741, 1.0812502916125, 1.24921880953755, 1.3468750719818, 1.50312510545521, 1.88984392539134, 2.37812526369453, 2.39375026652023, 2.36640651157525, 2.38593776510738, 2.62031280749291, 2.69843782162142, 2.85078159917202, 2.90546910906198), lat = c(1.046062476682306, 1.05624976964876, 1.03671851611663, 1.13828103448369, 1.16562478942867, 1.26718730779574, 1.43124983746561, 1.32968731909855, 1.32187481768569, 1.36624981485999, 1.28281231062144, 1.20468729649293, 1.13828103448369, 1.08749977503016, 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
```
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

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

Description

Provide square polygons or their node coordinates along a segment

Usage

```r
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. It has a plot method and can straightfully be handled by writeShapePoly (see readShapePoly) of the maptools library to write a shapefile. 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.

Author(s)

Patrick Giraudoux <pgiraudo@univ-fcomte.fr>

See Also

SpatialPolygonsDataFrame-class, readShapePoly, overlay, 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]],border="blue")
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)

---

**pclip**

*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

`pclip(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
permcont

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 columns 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)

## Not run:
## 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=250)

library(nlme)
fm2 <- lme(distance ~ age + Sex, data = Orthodont, random = ~ 1)
PermTest(fm2,B=250)

## End(Not run)

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


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 specified with probs
**polycirc**

**Value**

a vector of the two CI limits

**Author(s)**

Patrick Giraudoux <pgiraudo@univ-fcomte.fr>

**See Also**

piankabio

**Examples**

```r
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**

```r
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.
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 (default 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

```r
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)
```

---

### Description

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

### Usage

```r
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

### Author(s)

Patrick Giraudoux, patrick.giraudoux@univ-fcomte.fr

### See Also

- `text`
Examples

```r
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 Mansour Amroun 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

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

**Description**  
print method for objects of class 'mc'

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

---

**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**  
```r  
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 reads a raster file using GDAL within the bounding box of a spatial object. 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", seg=FALSE, 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, or others provided in gpsbabel
seg track ID type: FALSE for numbers, TRUE for GPS track IDs
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 successfully with "/dev/ttyUSB0"; on Windows commonly "com4:" or similar.
Value

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.

References

http://www.gpsbabel.org

See Also

readGPS

Examples

```r
## Not run:
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`
rwhatbufCat

Examples

toremove<-NULL
ls()
## Not run:
rmls()# select the object 'toremove' and click OK

## End(Not run)
ls()

---

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

Description

Analyses 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 centered
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 centered on point sites.

Value

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

See Also

rwhatbufNum, rwhatpoly, 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
mymaps <- spsample(meuse.grid, n=10, type="random") # random points are appx 10, see spsample doc
mymaps <- SpatialPointsDataFrame(coordinates(mymaps), data.frame(id=1:nrow(mymaps@coords)))

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

rwhatbufCat2 Analyses the contents of a raster file readable with rgdal of categorical values within various buffer sizes centred on points

Description

Analyses the contents of a raster file readable with rgdal of categorical values within various buffer sizes centred on 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 the 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 \texttt{rwhatbufCat}. Empty buffer (no pixel inside) gives (so far) an avoidable topology and dimension errors at reading and stop computation. This can be avoided adjusting buffer size so that the smaller buffer size includes at least one pixel in every position.

Value

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

See Also

\texttt{rwhatbufNum, rwhatpoly, rwhatbufCat}

Examples

library(sp)

myrasternamexystem.file("pictures/SP27GTIF.TIF", package = "rgdal")[1]

mylocationsx-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

resultxrwhatbufCat2(myrasternamexylocationsx, c(500, 1000))

result

resultxrwhatbufCat2(myrasternamexylocationsx, c(500, 1000), asList=TRUE)

result[[1]] # results for buffer 500 (5 buffer centers)

result[[2]] # results for buffer 1000 (5 buffer centers)

---

\texttt{rwhatbufNum}  

\textit{Analyses the contents of a SpatialPixelsDataFrame or a SpatialGridDataFrame of numerical values within various buffer sizes centred on points}

Description

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

Usage

\texttt{rwhatbufNum(rast, sites, bufsizes, att=1)}
Arguments

- **rast** object of class `SpatialPixelsDataFrame` or `SpatialGridDataFrame` to analyse
- **sites** object of class `SpatialPointsDataFrame` containing the 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

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 point site within the buffer

See Also

`rwhatbufCat`, `rwhatpoly`, `rwhatbufCat2`

Examples

```r
library(pgirmess)
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)
```
rwhatpoly

Analyses the contents of a SpatialPixelsDataFrame or a SpatialGridDataFrame within polygons

**Description**
Analyses the contents of a SpatialPixelsDataFrame or a SpatialGridDataFrame within polygons

**Usage**
rwhatpoly(sgdf, SP, att=1, NAin=TRUE, num=FALSE)

**Arguments**
- **sgdf**: object of class `SpatialPixelsDataFrame` or `SpatialGridDataFrame` to analyse
- **SP**: object of class `SpatialPolygons`
- **att**: column number of the attribute variable
- **NAin**: should NA values within polygons be counted (default to yes = TRUE)
- **num**: TRUE if pixel values are numeric

**Details**
If num is FALSE (the default) generates a table with the frequency of each category of the raster (`SpatialGridDataFrame`) within each polygon of the `SpatialPolygons` object.
If num is TRUE, it generates a list in which items are the values of the pixels included in each polygon

**Value**
If num if FALSE, a table. Each row is a polygon count (row name = polygon ID number), the last one the count of all values out of any polygons; each column is a raster category
If num is TRUE, a list. Each item is a vector of the pixel values included in the corresponding polygon

**See Also**
- `rwhatbufCat`, `rwhatbufNum`, `overlay`

**Examples**

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

# SpatialPolygons creation
polylist <-
list(structure(c(180016, 180225, 180533, 180615, 180588, 180452, 
180034, 180016, 332182, 332319, 332518, 332418, 332074, 331774, 
331756, 332182), .Dim = c(8L, 2L)), structure(c(179907, 180325, 
180397, 180152, 179781, 179672, 179735, 179907, 331611, 331611, 
331266, 330931, 330967, 331266, 331466, 331611), .Dim = c(8L, 
2L)), structure(c(179499, 179971, 180343, 180161, 179753, 179418, 
179499, 330577, 330768, 330468, 330096, 330078, 330369, 330577 
), .Dim = c(7L, 2L)))

SP<-SpatialPolygons(list(Polygons(list(Polygon(polylist[[1]])),ID="P1"), 
Polygons(list(Polygon(polylist[[2]])),ID="P2"),Polygons(list(Polygon(polylist[[3]])), 
ID="P3"),p0=as.integer(c(1,2,3)))

# raster and polygons display
image(meuse.grid,att=4,col=c("red","green","blue")) # soil map
plot(SP,add=TRUE)

text(coordinates(SP),sapply(SP@polygons,function(x) x@ID))

# extraction
# get the number of pixels of each soil category (column) in each polygon (rows)
rwhatpoly(meuse.grid,SP,att=4)
# get the number of pixels of each soil category (column), NAs excluded, in each polygon (rows)
rwhatpoly(meuse.grid,SP,att=4,NAin=FALSE)

image(meuse.grid,att=3) # distance map
plot(SP,add=TRUE)

text(coordinates(SP),sapply(SP@polygons,function(x) x@ID))

# get the number of pixels of each soil category (column), NAs excluded,
# in each polygon (rows), the output is a list (lapply or sapply can be used to get statistics)
vals<-rwhatpoly(meuse.grid,SP,att=3,num=TRUE)
names(vals)<-c("P1","P2","P3",NA) # NA for pixels that are outside of any polygon
vals[[1]]
sapply(vals,mean,na.rm=TRUE)
```

**Description**

Draw line segments between pairs of points from a vector, matrix or data frame of 4 points coordinates \(x_0, y_0, x_1, y_1\).
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", ...)

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[1:3],myvec[2:4],type="n",xlab="",ylab="")
Segments(myvec)

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

- **aModel**: A lm or glm model or a list of relevant models (see details)
- **dropNull**: If TRUE, drops the simplest model (e.g. \( y \sim 1 \))
- **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 \( y \sim 1 \) 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, dpleydel@univ-fcomte.fr
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
shannonbio

See Also

shannonbio

Examples

```r
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

**Author(s)**

Patrick Giraudoux <pgiraudo@univ-fcomte.fr>

**See Also**

`shannon`, `difshannonbio`
shannonbioboot

Examples

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

Description

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

Usage

```r
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

Bootstrap 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*)

Author(s)

Patrick Giraudoux <pgiraudo@univ-fcomte.fr

See Also

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

Examples

```r
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'
```
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

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

Author(s)

Patrick Giraudoux <pgiraudo@univ-fcomte.fr>

Examples

```r
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

```r
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 closest point and removes all points within the USR, and so on...
Value
A \textbf{SpatialPoints} object of the track thinned.

See Also
\texttt{mergeTrackObs}

Examples

```r
library(sp)
mySPDF <- structure(list(x = c(748775, 748807, 748834, 748854, 748871,
                           748873, 748880, 748910, 748919, 748917, 748921, 748923, 748924,
                           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, 104900, 104956, 104947,
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)
```

---

\textbf{trans2pix} \hspace{1cm} \textit{Convert a transect coordinate file with some landmarks into a matrix with intermediate coordinates.}

\textbf{Description}
Convert a transect coordinate file with some landmarks and NA values in between into a matrix with intermediate coordinates.

\textbf{Usage}
\texttt{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 of each lines materialised with NA values.

Value

A matrix with the intermediate coordinates computed.

See Also

trans2seg

Examples

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

A matrix of 4 columns to be passed eg to fonctions as "segments".

See Also

trans2pix

Examples

```r
x<-c(10, NA, NA, 56, NA, NA, 100)
y<-c(23, 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 points at regular distance along the lines

Description

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

Usage

```r
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 any track line (roads, paths, transects, etc.) into series of regular points. Each point may be though of as corresponding to the centre of one interval

Value

A SpatialPointsDataFrame

See Also

trans2pix, thintrack, mergeTrackObs
TukeyHSDs

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))
  S1l = Line(l1)
  S1la = Line(l1a)
  S12 = Line(l2)
  S1 = Lines(list(S1l, S1la), ID="a")
  S2 = Lines(list(S12), ID="b")
  S1 = SpatialLines(list(S1, S2))
  plot(S1, col = c("red", "blue"))

  trpt<-transLines2pix(S1,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
uploadGPS

Examples

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

### Description

Upload waypoints to Garmin GPS, using gpsbabel

### Usage

```r
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](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

```r
## Not run:
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**

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

Description

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

Usage

```r
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

Author(s)

Patrick Giraudoux, pgiraudo@univ-fcomte.fr

See Also

`symbols, mean, median, scale`
valchisq

Examples

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

z <- val4sym(rnorm(30))
symbols(x, y, circle = z$s$size, inches = 0.2, bg = z$col)

z <- val4sym(scale(rnorm(30)))
symbols(x, y, circle = z$s$size, inches = 0.2, bg = z$col)

z <- val4sym(rnorm(30), col = c("green", "violet"))
symbols(x, y, circle = z$s$size, inches = 0.2, bg = z$col)

z <- val4sym(rnorm(30), trim = 0.025)
symbols(x, y, circle = z$s$size, inches = 0.2, bg = z$col)

z <- val4sym(rnorm(30), median)
symbols(x, y, circle = z$s$size, inches = 0.2, bg = z$col)

myfunc <- function(x) 20 # passes an arbitrary constant
z <- val4sym(1:30, myfunc)
symbols(x, y, circle = z$s$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

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

Description

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

Usage

```r
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

An ascii text file, tab delimited.
writeGPX

Author(s)
Patrick Giraudoux <pgiraudo@univ-fcomte.fr>

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

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

```r
coords<-data.frame(ID=c("C18J01", "C18J02"),Long= c(-46.996602, 47.002745),
Lat=c(-6.148734, 6.148299),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(sobj, filename)

Arguments
sobj 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

```r
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

*Topic IO
  gps2gpx, 18
  readGDALbbox, 35
  readVista, 36
  uploadGPS, 55
  writeGPX, 59
  writePRJ, 60

*Topic array
  pclig, 26
  tabcont2categ, 49
  valchisq, 57

*Topic color
  val14sym, 56

*Topic connection
  gps2gpx, 18
  readGDALbbox, 35
  readVista, 36
  uploadGPS, 55
  writeGPX, 59
  writePRJ, 60

*Topic datasets
  preybiom, 34
  siegelp179, 49

*Topic distribution
  permcont, 27

*Topic dplot
  diag2edge, 9
  pave, 24
  polycirc2, 32
  postxt, 33
  val14sym, 56

*Topic hplot
  pairsrp, 23
  Segments, 43

*Topic htest
  CI, 3
  cormat, 5
  friedmanmc, 17
  kruskalmc, 19
  ks.gof, 20
  PermTest, 28
  piankabioboot, 30
  shannonbioboot, 48
  TukeyHSDs, 54

*Topic manip
  expandpoly, 16
  polycirc, 31

*Topic misc
  classnum, 4
  clipQGIS2df, 5
  date2winter, 8
  difshannonbio, 10
  piankabio, 29
  shannon, 46
  shannonbio, 47

*Topic models
  selMod, 44

*Topic print
  print.mc, 35

*Topic spatial
  correlog, 6
  diag2edge, 9
  dirProj, 11
  dirSeg, 12
  distNNeigh, 13
  distNode, 13
  distSeg, 14
  distTot, 15
  pave, 24
  rwhatbufCat, 38
  rwhatbufCat2, 39
  rwhatbufNum, 40
  rwhatpoly, 42
  thintrack, 50

*Topic utilities, spatial
  mergeTrackObs, 21
  trans2pix, 51
  transLines2pix, 53
<table>
<thead>
<tr>
<th>Topic</th>
<th>utilities</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mean, 56</td>
</tr>
<tr>
<td></td>
<td>median, 56</td>
</tr>
<tr>
<td></td>
<td>mergeTrackObs, 21, 51, 53</td>
</tr>
<tr>
<td></td>
<td>moran.test, 7</td>
</tr>
<tr>
<td></td>
<td>nbdists, 13</td>
</tr>
<tr>
<td></td>
<td>overlay, 25, 42</td>
</tr>
<tr>
<td></td>
<td>pairs, 24</td>
</tr>
<tr>
<td></td>
<td>pairsrp, 23</td>
</tr>
<tr>
<td></td>
<td>pave, 9, 24</td>
</tr>
<tr>
<td></td>
<td>pclig, 26</td>
</tr>
<tr>
<td></td>
<td>permcont, 27</td>
</tr>
<tr>
<td></td>
<td>PermTest, 28</td>
</tr>
<tr>
<td></td>
<td>piankabio, 29, 31</td>
</tr>
<tr>
<td></td>
<td>piankabioboot, 30, 30</td>
</tr>
<tr>
<td></td>
<td>plot.correlog (correlog), 6</td>
</tr>
<tr>
<td></td>
<td>polycirc, 31, 32</td>
</tr>
<tr>
<td></td>
<td>polycirc2, 32</td>
</tr>
<tr>
<td></td>
<td>polygon, 16, 32</td>
</tr>
<tr>
<td></td>
<td>postxt, 33</td>
</tr>
<tr>
<td></td>
<td>preybiom, 34</td>
</tr>
<tr>
<td></td>
<td>print.clnum (classnum), 4</td>
</tr>
<tr>
<td></td>
<td>print.correlog (correlog), 6</td>
</tr>
<tr>
<td></td>
<td>print.mc, 35</td>
</tr>
<tr>
<td></td>
<td>print.PermTest (PermTest), 28</td>
</tr>
<tr>
<td></td>
<td>proj4string, 60</td>
</tr>
<tr>
<td></td>
<td>prop.table, 27</td>
</tr>
<tr>
<td></td>
<td>prop.test, 3</td>
</tr>
<tr>
<td></td>
<td>readGDAL, 35, 36</td>
</tr>
<tr>
<td></td>
<td>readGDALbbox, 35</td>
</tr>
<tr>
<td></td>
<td>readGPS, 37</td>
</tr>
<tr>
<td></td>
<td>readOGR, 18, 60</td>
</tr>
<tr>
<td></td>
<td>readShapePoly, 25</td>
</tr>
<tr>
<td></td>
<td>readVista, 36</td>
</tr>
<tr>
<td></td>
<td>relevel, 20</td>
</tr>
<tr>
<td></td>
<td>rmls, 37</td>
</tr>
<tr>
<td></td>
<td>rwhatbufCat, 38, 40–42</td>
</tr>
<tr>
<td></td>
<td>rwhatbufCat2, 38, 39, 41</td>
</tr>
<tr>
<td></td>
<td>rwhatbufNum, 38, 40, 40, 42</td>
</tr>
<tr>
<td></td>
<td>rwhatpoly, 38, 40, 41, 42</td>
</tr>
<tr>
<td></td>
<td>scale, 56</td>
</tr>
<tr>
<td></td>
<td>Segments, 43</td>
</tr>
<tr>
<td></td>
<td>segments, 44</td>
</tr>
<tr>
<td></td>
<td>selMod, 44</td>
</tr>
<tr>
<td></td>
<td>shannon, 46, 47</td>
</tr>
</tbody>
</table>
shannonbio, 10, 47, 47, 48
shannonbioboot, 48
siegelp179, 49
SpatialGridDataFrame, 38, 41, 42
SpatialLines, 53
SpatialLinesDataFrame, 53
SpatialPixelsDataFrame, 38, 41, 42
SpatialPoints, 22, 38, 39, 51
SpatialPointsDataFrame, 22, 41, 50, 53
SpatialPolygons, 42
symbols, 56

tabcont2categ, 49
text, 33
thintrack, 22, 50, 53
trans2pix, 51, 53
trans2seg, 52, 52
transLines2pix, 22, 53
TukeyHSD, 54
TukeyHSDs, 54

uploadGPS, 18, 55

val4symb, 56
valchisq, 57

write.delim, 58
write.table, 59
writeGPX, 55, 59
writeOGR, 60
writePRJ, 60