Package 'SDMTools'

February 19, 2015

Type Package

Title Species Distribution Modelling Tools: Tools for processing data associated with species distribution modelling exercises

Version 1.1-221

Date 2014-08-05

Author Jeremy VanDerWal, Lorena Falconi, Stephanie Januchowski, Luke Shoo and Collin Storlie

Maintainer Jeremy VanDerWal <jjvanderwal@gmail.com>

Imports R.utils

Suggests adehabitat, raster, sp

Description This packages provides a set of tools for post processing the outcomes of species distribution modeling exercises. It includes novel methods for comparing models and tracking changes in distributions through time. It further includes methods for visualizing outcomes, selecting thresholds, calculating measures of accuracy and landscape fragmentation statistics, etc.. This package was made possible in part by financial support from the Australian Research Council & ARC Research Network for Earth System Science.

License GPL (>= 3)

URL http://www.rforge.net/SDMTools/

LazyLoad yes

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-08-05 12:16:01

R topics documented:

`accuracy` *Measures of Model Accuracy*

Description

`accuracy` estimates six measures of accuracy for presence-absence or presence-psuedoabsence data. These include AUC, ommission rates, sensitivity, specificity, proportion correctly identified and Kappa.

Note: this method will exclude any missing data.

Usage

```
accuracy(obs, pred, threshold = 0.5)
```

Arguments

obs	a vector of observed values which must be 0 for absences and 1 for occurrences
pred	a vector of the same length as obs representing the predicted values. Values must be between 0 & 1 prepresenting a likelihood.
threshold	this can be: a) a single value representing a single threshold between 0 & 1; b) a vector of threshold values between 0 & 1; OR c) an integer value representing the number of equal interval threshold values between 0 & 1

Value

a data.frame with seven columns:

threshold	the threshold values representing each row of data
AUC	the AUC given the defined threshold value
ommission.rate	the ommission rate as a proportion of true occurrences misidentified given the defined threshold value
sensitivity	the sensitivity given the defined threshold value
specificity	the specificity given the defined threshold value
prop.correct	the proportion of the presence and absence records correctly identified given the defined threshold value
Kappa	the Kappa statistic of the model given the defined threshold value

Author(s)

Jeremy VanDerWal <jjvanderwal@gmail.com>

See Also

auc, Kappa, omission, sensitivity, specificity, prop.correct, confusion.matrix

Examples

```
#create some data
obs = c(sample(c(0,1),20,replace=TRUE),NA); obs = obs[order(obs)]
pred = runif(length(obs),0,1); pred = pred[order(pred)]

#calculate accuracy of the model with a single threshold value
accuracy(obs,pred,threshold=0.5)

#calculate accuracy given several defined thresholds
accuracy(obs,pred,threshold=c(0.33,0.5,0.66))

#calculate accuracy given a number of equal interval thresholds
accuracy(obs,pred,threshold=20)
```

`asc.from.raster` *Raster conversion functions for adehabitat, raster and sp packages*

Description

`asc.from.raster` and `asc.from.sp` extracts data from objects of class 'RasterLayer' (raster package) and class 'SpatialGridDataFrame' (sp package) into an object of class 'asc' (SDMTools & adehabitat packages).

`raster.from.asc` and `sp.from.asc` does the reverse.

`as.asc` creates an object of class 'asc' (SDMTools & adehabitat packages) from a matrix of data. Code & helpfile associated with `as.asc` were modified from adehabitat package.

Usage

```
asc.from.raster(x)

raster.from.asc(x, projs = NA)

asc.from.sp(x)

sp.from.asc(x, projs = CRS(as.character(NA)))

as.asc(x, xll = 1, yll = 1, cellsize = 1, type = c("numeric", "factor"),
  lev = levels(factor(x)))
```

Arguments

`x`	is an object of class 'asc', 'RasterLayer' or 'SpatialGridDataFrame'. For the function `as.asc`, a matrix
`projs`	is a CRS projection string of the Proj4 package
`xll`	the x coordinate of the center of the lower left pixel of the map
`yll`	the y coordinate of the center of the lower left pixel of the map
`cellsize`	the size of a pixel on the studied map
`type`	a character string. Either `"numeric"` or `"factor"`
`lev`	if `type = "factor"`, either a vector giving the labels of the factor levels, or the name of a file giving the correspondence table of the map (see adehabitat as.asc helpfile details)

Details

These functions provide capabilities of using scripts / functions from many packages including adehabitat (plus e.g, SDMTools), sp (plus e.g., maptools, rgdal) and raster.

Value

Returns an object of class requested.

Author(s)

Jeremy VanDerWal <jjvanderwal@gmail.com>

Examples

```
#create a simple object of class 'asc'
tasc = as.asc(matrix(rep(x=1:10, times=1000),nr=100)); print(tasc)
str(tasc)

#convert to RasterLayer
traster = raster.from.asc(tasc)
str(traster)

#convert to SpatialGridDataFrame
tgrid = sp.from.asc(tasc)
str(tgrid)

#create a basic object of class asc
tasc = as.asc(matrix(rep(x=1:10, times=1000),nr=100)); print(tasc)
```

---

asc2dataframe *Ascii Grid Files to Dataframe and Dataframe to Ascii Grid Files*

Description

asc2dataframe converts a list of Esri ascii grid formatted files to a data.frame consisting of only locations with data.

dataframe2asc converts a data.frame or matrix with spatial data to Esri ascii grid formatted files.

Usage

```
asc2dataframe(filenames, varnames = NULL, gz = FALSE)

dataframe2asc(tdata, filenames = NULL, outdir = getwd(), gz = FALSE)
```

Arguments

filenames	is a vector of file names
varnames	is a vector of names for the output columns, and must be the same length as files
tdata	is the data.frame which has y, x coordinates (OR lat,lon) and columns for the data to be output (MUST be in that order)
outdir	is the output directory, the default is the current working directory
gz	boolean defining if the ascii grid files are gzip compressed

Details

asc2dataframe: The ascii grid files can be read in gzip compress format. The dataframe returned contains the X and Y coordinate columns followed by columns of data.

dataframe2asc: If filenames is null, column names will be used. The data.frame has to contain the Y and X coordinates and the data as columns. The ascii grid files can be created as gzip compress format and would be saved in the outdir.

Value

`asc2dataframe`	Returns a dataframe with XY coordinates and the data of each ascii grid files, as columns.
`dataframe2asc`	Returns an asc grid file for each data column within the data.frame.

Author(s)

Lorena Falconi <lorefalconi@gmail.com>

Examples

```
#Create 2 ascii files
y=seq(10,50,0.5)
x=seq(140,180,0.5)
cellsize=0.5
data1=sample(160,140)
data2=sample(158,140)
out1.asc=as.asc(matrix(data1,nc=y, nr=x), xll=min(x), yll=min(y), cellsize=cellsize)
out2.asc=as.asc(matrix(data2,nc=y, nr=x), xll=min(x), yll=min(y), cellsize=cellsize)
#write the ascii files to the work directory
write.asc(out1.asc, 'out1.asc')
write.asc(out2.asc, 'out2.asc')
#list the ascii files
ascfiles=c('out1.asc', 'out2.asc')
#generate a dataframe from the ascii files
tdata=asc2dataframe(ascfiles)
tdata

#remove the files
unlink('out1.asc'); unlink('out2.asc')

#convert the dataframe tdata to ascii grid files
dataframe2asc(tdata)

#remove the files
unlink('var.1.asc'); unlink('var.2.asc')
```

auc *Area Under the Curve of the Reciever Operating Curve*

Description

auc estimates the AUC of the ROC using a Mann-Whitney U statistic.

Note: this method will exclude any missing data.

Usage

```
auc(obs, pred)
```

Arguments

obs	a vector of observed values which must be 0 for absences and 1 for occurrences
pred	a vector of the same length as obs representing the predicted values. Values must be between 0 & 1 representing a likelihood.

Value

Returns a single value represting the AUC value.

Author(s)

Jeremy VanDerWal <jjvanderwal@gmail.com>

See Also

Kappa, omission, sensitivity, specificity, prop.correct, confusion.matrix, accuracy

Examples

```
#create some data
obs = c(sample(c(0,1),20,replace=TRUE),NA)
pred = runif(length(obs),0,1)

#calculate AUC from the random data
auc(obs,pred)

#calculate an example 'perfect' AUC
obs = obs[order(obs)]
pred = pred[order(pred)]
auc(obs,pred)
```

`circular.averaging` *Circular Averaging based on Vector Averaging*

Description

`circular.averaging` calculates the average direction (0 - 360) given a vector of directions.

`vector.averaging` calculates the average distance and direction given a vector of directions and a vector of distances.

Usage

```
circular.averaging(direction, deg = TRUE)

vector.averaging(direction, distance, deg = TRUE)
```

Arguments

`direction`	a vector of directions given in degrees (0 - 360) if `deg==TRUE` or in radians if `deg==FALSE`
`distance`	a vector of distances associated with each direction
`deg`	a boolean object defining if `direction` is in degrees (TRUE) or radians (FALSE)

Details

functions return NA if the average distance or direction is not valid... e.g., when averaging directions of 0 & 180 degrees, the result could theoretically be 90 or 270 but is practically neither.

Value

`circular.averaging` returns the average direction while `vector.averaging` returns a list with 2 elements distance & direction

Author(s)

Jeremy VanDerWal <jjvanderwal@gmail.com> & Lorena Falconi <lorefalconi@gmail.com>

Examples

```
#EXAMPLE circular.averaging
circular.averaging(c(0,90,180,270)) #result is NA
circular.averaging(c(70,82,96,110,119,259))

#EXAMPLE vector.averaging
vector.averaging(c(10,20,70,78,108), distance=10)
vector.averaging(c(159,220,258,273,310),distance=runif(5))
```

ClassStat *Landscape Class Statistics*

Description

ClassStat calculates the class statistics for patch types identified in a matrix of data or in a raster of class 'asc' (SDMTools & adehabitat packages), 'RasterLayer' (raster package) or 'SpatialGridDataFrame' (sp package).

Usage

```
ClassStat(mat, cellsize = 1, bkgd = NA, latlon = FALSE)
```

Arguments

mat	a matrix of data with patches identified as classes (unique integer values) as e.g., a binary lanscape of a species distribution or a vegetation map. Matrix can be a raster of class 'asc' (adehabitat package), 'RasterLayer' (raster package) or 'SpatialGridDataFrame' (sp package)
cellsize	cell size (in meters) is a single value representing the width/height of cell edges (assuming square cells)
bkgd	the background value for which statistics will not be calculated
latlon	boolean value representing if the data is geographic. If latlon == TRUE, matrix must be of class 'asc', 'RasterLayer' or 'SpatialGridDataFrame'

Details

The class statistics are based on statistics calculated by fragstats http://www.umass.edu/landeco/research/fragstats/fragstats.html.

Value

a data.frame listing

class	a particular patch type from the original input matrix (mat).
n.patches	the number of patches of a particular patch type or in a class.
total.area	the sum of the areas (m2) of all patches of the corresponding patch type.
prop.landscape	the proportion of the total lanscape represented by this class
patch.density	the numbers of patches of the corresponding patch type divided by total landscape area (m2).
total.edge	the total edge length of a particular patch type.
edge.density	edge length on a per unit area basis that facilitates comparison among landscapes of varying size.

`landscape.shape.index`
: a standardized measure of total edge or edge density that adjusts for the size of the landscape.

`largest.patch.index`
: largest patch index quantifies the percentage of total landscape area comprised by the largest patch.

`mean.patch.area`
: average area of patches.

`sd.patch.area` standard deviation of patch areas.

`min.patch.area` the minimum patch area of the total patch areas.

`max.patch.area` the maximum patch area of the total patch areas.

`perimeter.area.frac.dim`
: perimeter-area fractal dimension equals 2 divided by the slope of regression line obtained by regressing the logarithm of patch area (m2) against the logarithm of patch perimeter (m).

`mean.perim.area.ratio`
: the mean of the ratio patch perimeter. The perimeter-area ratio is equal to the ratio of the patch perimeter (m) to area (m2).

`sd.perim.area.ratio`
: standard deviation of the ratio patch perimeter.

`min.perim.area.ratio`
: minimum perimeter area ratio

`max.perim.area.ratio`
: maximum perimeter area ratio.

`mean.shape.index`
: mean of shape index

`sd.shape.index` standard deviation of shape index.

`min.shape.index`
: the minimum shape index.

`max.shape.index`
: the maximum shape index.

`mean.frac.dim.index`
: mean of fractal dimension index.

`sd.frac.dim.index`
: standard deviation of fractal dimension index.

`min.frac.dim.index`
: the minimum fractal dimension index.

`max.frac.dim.index`
: the maximum fractal dimension index.

`total.core.area`
: the sum of the core areas of the patches (m2).

`prop.landscape.core`
: proportional landscape core

`mean.patch.core.area`
: mean patch core area.

`sd.patch.core.area`
: standard deviation of patch core area.

`min.patch.core.area`
: the minimum patch core area.

`max.patch.core.area`
: the maximum patch core area.

`prop.like.adjacencies`
: calculated from the adjacency matrix, which shows the frequency with which different pairs of patch types (including like adjacencies between the same patch type) appear side-by-side on the map (measures the degree of aggregation of patch types).

`aggregation.index`
: computed simply as an area-weighted mean class aggregation index, where each class is weighted by its proportional area in the landscape.

`lanscape.division.index`
: based on the cumulative patch area distribution and is interpreted as the probability that two randomly chosen pixels in the landscape are not situated in the same patch

`splitting.index`
: based on the cumulative patch area distribution and is interpreted as the effective mesh number, or number of patches with a constant patch size when the landscape is subdivided into S patches, where S is the value of the splitting index.

`effective.mesh.size`
: equals 1 divided by the total landscape area (m2) multiplied by the sum of patch area (m2) squared, summed across all patches in the landscape.

`patch.cohesion.index`
: measures the physical connectedness of the corresponding patch type.

Author(s)

Jeremy VanDerWal <jjvanderwal@gmail.com>

References

McGarigal, K., S. A. Cushman, M. C. Neel, and E. Ene. 2002. FRAGSTATS: Spatial Pattern Analysis Program for Categorical Maps. Computer software program produced by the authors at the University of Massachusetts, Amherst. Available at the following web site: www.umass.edu/landeco/research/fragstats/fragstats.html

See Also

PatchStat, ConnCompLabel

Examples

```
#define a simple binary matrix
tmat = { matrix(c( 0,0,0,1,0,0,1,1,0,1,
                   0,0,1,0,1,0,0,0,0,0,
```

```
                    0,1,NA,1,0,1,0,0,0,1,
                    1,0,1,1,1,0,1,0,0,1,
                    0,1,0,1,0,1,0,0,0,1,
                    0,0,1,0,1,0,0,1,1,0,
                    1,0,0,1,0,0,1,0,0,1,
                    0,1,0,0,0,1,0,0,0,1,
                    0,0,1,1,1,0,0,0,0,1,
                    1,1,1,0,0,0,0,0,0,1),nr=10,byrow=TRUE) }

#do the connected component labelling
ccl.mat = ConnCompLabel(tmat)
ccl.mat
image(t(ccl.mat[10:1,]),col=c('grey',rainbow(length(unique(ccl.mat))-1)))

#calculate the patch statistics
ps.data = PatchStat(ccl.mat)
ps.data

#calculate the class statistics
cl.data = ClassStat(tmat)
cl.data

#identify background data is 0
cl.data = ClassStat(tmat,bkgd=0)
cl.data
```

`COGravity` *Centre of Gravity or Mass calculations for spatial data*

Description

`COGravity` calculates the Centre of Gravity (or also known as Centre of Mass) for point or raster spatial data.

Note: NA data is automatically ommitted from analysis.

Usage

```
COGravity(x, y = NULL, z = NULL, wt = NULL)
```

Arguments

`x`	a vector of e.g., longitudes or eastings, or a raster of class 'asc', 'RasterLayer' or 'SpatialGridDataFrame'.
`y`	a vector of e.g., latitudes or northings.
`z`	a vector of e.g., elevations.
`wt`	a vector or raster of class 'asc', 'RasterLayer' or 'SpatialGridDataFrame' representing weights for data.

Details

For raster-based data, if `wt` is missing, the values of the ascii are assumed to be the weights; otherwise, the values are assumed to be the `z` values.

Value

Returns a named vector of data representing the Centre of Gravity in x, y & z dimensions (depending on data supplied).

Author(s)

Jeremy VanDerWal <jjvanderwal@gmail.com>

Examples

```
#create some points
x = seq(154,110,length=25)
y = seq(-10,-54,length=25)
z = seq(100,200,length=25)
wt = runif(25) #random weights
#calculate the Centre of Gravity for these points
COGravity(x,y,z,wt)

#create a simple objects of class 'asc'
x = as.asc(matrix(1:50,nr=50,nc=50))
wt = as.asc(matrix(runif(50),nr=50,nc=50))

#calculate COG with weighting defined in x
COGravity(x)
#calculate COG with weighting defined in wt (values in x are assumed elevation (z))
COGravity(x,wt=wt)
```

---

compare.matrix *Biplot Comparison of Matrices*

Description

`compare.matrix` compares the values within two matrices (e.g., ESRI ArcInfo ASCII raster files) and produces a biplot that shows the frequency of each data combination shared between the matrices. The plot is overlayed with contour lines that demarcate parts of the the plot that share the same frequency of data combinations.

NOTE: it is assumed the matrices are of the same extent, cell size and scaled to be the same units.

Usage

```
compare.matrix(x, y, nbins, ...)
```

Arguments

x	a matrix of data; the matrix can be a raster of class 'asc' (adehabitat package), 'RasterLayer' (raster package) or 'SpatialGridDataFrame' (sp package)
y	a matrix of data of the same extent, cell size and class as 'x'
nbins	number of equally spaced bins used to partition range of values in 'x' & 'y'
...	other graphical parameters defined by image(), contour(), or plot()

Value

Nothing is returned but images are created.

Author(s)

Luke Shoo <luke.shoo@jcu.edu.au>

Examples

```
#create some simple objects of class 'asc'
tasc = as.asc(matrix(rep(x=1:10, times=1000),nr=100)); print(tasc)
#modify the asc objects so that they are slightly different
tasc1 = tasc + runif(n = 10000, min = -1, max = 1)
tasc2 = tasc + rnorm(n = 10000, mean = 1, sd = 1)

#create some images
#basic plot showing the density of data combinations shared
#by the two matrices
compare.matrix(tasc1,tasc2,20)

#same as previous but with data partioned amoung more bins
compare.matrix(tasc1,tasc2,50)

#same as previous but altering the number of contour levels
#and adding more graphical functions
compare.matrix(tasc1,tasc2,50,nlevels=5, xlab='asc1',ylab='asc2',
  main='Comparison between asc and asc2', bg="grey")
```

---

confusion.matrix *Confusion Matrix*

Description

confusion.matrix calculates a confusion matrix.

Note: this method will exclude any missing data

Usage

```
confusion.matrix(obs, pred, threshold = 0.5)
```

Arguments

obs	a vector of observed values which must be 0 for absences and 1 for occurrences
pred	a vector of the same length as obs representing the predicted values. Values must be between 0 & 1 prepresenting a likelihood.
threshold	a single threshold value between 0 & 1

Value

Returns a confusion matrix (table) of class 'confusion.matrix' representing counts of true & false presences and absences.

Author(s)

Jeremy VanDerWal <jjvanderwal@gmail.com>

See Also

auc, Kappa, omission, sensitivity, specificity, prop.correct, accuracy

Examples

```
#create some data
obs = c(sample(c(0,1),20,replace=TRUE),NA); obs = obs[order(obs)]
pred = runif(length(obs),0,1); pred = pred[order(pred)]

#calculate the confusion matrix
confusion.matrix(obs,pred,threshold=0.5)
```

ConnCompLabel *Connected Components Labelling – Unique Patch Labelling*

Description

ConnCompLabel is a 1 pass implementation of connected components labelling. Here it is applied to identify disjunt patches within a distribution.

The raster matrix can be a raster of class 'asc' (adehabitat package), 'RasterLayer' (raster package) or 'SpatialGridDataFrame' (sp package).

Usage

```
ConnCompLabel(mat)
```

Arguments

mat	is a binary matrix of data with 0 representing background and 1 representing environment of interest. NA values are acceptable. The matrix can be a raster of class 'asc' (this & adehabitat package), 'RasterLayer' (raster package) or 'SpatialGridDataFrame' (sp package)

Value

A matrix of the same dim and class of `mat` in which unique components (individual patches) are numbered 1:n with 0 remaining background value.

Author(s)

Jeremy VanDerWal <jjvanderwal@gmail.com>

References

Chang, F., C.-J. Chen, and C.-J. Lu. 2004. A linear-time component-labeling algorithm using contour tracing technique. Comput. Vis. Image Underst. 93:206-220.

See Also

PatchStat, ClassStat

Examples

```
#define a simple binary matrix
tmat = { matrix(c( 0,0,0,1,0,0,1,1,0,1,
                   0,0,1,0,1,0,0,0,0,0,
                   0,1,NA,1,0,1,0,0,0,1,
                   1,0,1,1,1,0,1,0,0,1,
                   0,1,0,1,0,1,0,0,0,1,
                   0,0,1,0,1,0,0,1,1,0,
                   1,0,0,1,0,0,1,0,0,1,
                   0,1,0,0,0,1,0,0,0,1,
                   0,0,1,1,1,0,0,0,0,1,
                   1,1,1,0,0,0,0,0,0,1),nr=10,byrow=TRUE) }

#do the connected component labelling
ccl.mat = ConnCompLabel(tmat)
ccl.mat
image(t(ccl.mat[10:1,]),col=c('grey',rainbow(length(unique(ccl.mat))-1)))
```

---

destination *Vincenty Direct Calculation of a Destination*

Description

`destination` estimates the destination latitude and longitude given a starting latitude and longitude, a bearing and distance.

For general information on Vincenty's formula, see e.g., http://en.wikipedia.org/wiki/Vincenty's_formulae. It states:
Vincenty's formulae are two related iterative methods used in geodesy to calculate the distance between two points on the surface of an spheroid, developed by Thaddeus Vincenty in 1975. They

are based on the assumption that the figure of the Earth is an oblate spheroid, and hence are more accurate than methods such as great-circle distance which assume a spherical Earth.

Note: this method assumes a locations are lat & lon given in WGS 84.

Usage

```
destination(lat, lon, bearing, distance)
```

Arguments

`lat`	a single value or vector of values representing latitude in decimal degrees from -90 to 90 degrees.
`lon`	a single value or vector of values representing longitude in decimal degrees from -180 to 180 degrees.
`bearing`	a single value or vector of values representing the bearings (directions) of interest ranging from 0 to 360 degrees.
`distance`	a single value or vector of values representing the distances in metres to the destination.

Details

Typical useages are:

1. a single start location, bearing and distance to give a single output location
   –output would be a single destination location
2. a single start location with one or more bearings or distances to give multiple output locations
   –output would be a destination locations for each combination of bearings and distances
3. multiple start locations with a single bearing or distance
   –output would be a destination locations representing the bearing and distance from each of the start locations
4. multiple start locations with multiple bearings or distances
   –output would be a destination locations representing the combinations of bearings and distances from each of the start locations
   – NOTE that the bearing and distance vectors must be of the same length of the input lat and long.

See examples for all possible usages.

Value

Returns a data.frame with:

`lon1`	the original longitude
`lat1`	the original latitude
`bearing`	the bearing used
`distance`	the distance used

lon2	the destination longitude
lat2	the destination latitude

Author(s)

Jeremy VanDerWal <jjvanderwal@gmail.com>

Source

The source code here was modified from http://www.movable-type.co.uk/scripts/latlong-vincenty-direct.html.

Destinations were validated against Geoscience Australia calculations (http://www.ga.gov.au/geodesy/datums/vincenty_direct.jsp).

References

Vincenty, T. 1975. Direct and Inverse Solutions of Geodesics on the Ellipsoid with application of Nested Equations. Survey Review, vol XXII no 176. http://www.ngs.noaa.gov/PUBS_LIB/inverse.pdf

Examples

```
###single lat lons
lats = -85; lons = 165
#single bearing & single distance
destination(lats,lons,bearing=180,distance=500000)

#multiple bearings
destination(lats,lons,bearing=seq(0,360,length.out=9),distance=500000)

#multiple bearings
destination(lats,lons,bearing=45,distance=seq(0,5000000,length.out=11))

#multiple bearings, multiple distances
destination(lats,lons,bearing=seq(0,360,length.out=9),
    distance=seq(0,5000000,length.out=11))

###multiple lat lons
lats = seq(-90,90,length.out=9); lons = seq(-180,180,length.out=9)

#multiple lat lons but single bearings / distances
destination(lats,lons,bearing=45,distance=500000)

#different bearings for each lat lon
destination(lats,lons,bearing=seq(0,360,length.out=9),distance=500000)

#different distances for each lat lon
destination(lats,lons,bearing=45,distance=seq(0,5000000,length.out=9))

#different bearings & distances for each lat lon
```

```
destination(lats,lons,bearing=seq(0,360,length.out=9),
    distance=seq(0,5000000,length.out=9))
```

`distance` *Vincenty Direct Calculation of Distance and Direction*

Description

`distance` estimates the distance given a starting & ending latitude and longitude.

For general information on Vincenty's formula, see e.g., http://en.wikipedia.org/wiki/Vincenty's_formulae. It states:
Vincenty's formulae are two related iterative methods used in geodesy to calculate the distance between two points on the surface of an spheroid, developed by Thaddeus Vincenty in 1975. They are based on the assumption that the figure of the Earth is an oblate spheroid, and hence are more accurate than methods such as great-circle distance which assume a spherical Earth.

Note: this method assumes a locations are lat & lon given in WGS 84.

Direction, if requested, is the the initial bearing (sometimes referred to as forward azimuth) for which one would follow as a straight line along a great-circle arc from start to finish.

Note: this will fail if there are NA's in the data.

Usage

```
distance(lat1, lon1 = NULL, lat2 = NULL, lon2 = NULL, bearing = FALSE)
```

Arguments

`lat1`	a single value or vector of values representing latitude in decimal degrees from -90 to 90 degrees. Alternatively, a data.frame or matrix can be used here with each column representing lat1, lon1, lat2, lon2 (in that order).
`lon1`	a single value or vector of values representing longitude in decimal degrees from -180 to 180 degrees. If NULL, lat1 is assumed to be a matrix or data.frame.
`lat2`	a single value or vector of values representing latitude in decimal degrees from -90 to 90 degrees. If NULL, lat1 is assumed to be a matrix or data.frame.
`lon2`	a single value or vector of values representing longitude in decimal degrees from -180 to 180 degrees. If NULL, lat1 is assumed to be a matrix or data.frame.
`bearing`	boolean value as to calculate the direction as well as the distance.

Value

Returns a data.frame with:

`lon1`	the original longitude

lat1	the original latitude
lon2	the destination longitude
lat2	the destination latitude
distance	the distance used
bearing	if requested, the bearing between the two points

Author(s)

Jeremy VanDerWal <jjvanderwal@gmail.com>

Source

The source code for the distance algorithm here was modified from http://www.movable-type.co.uk/scripts/latlong-vincenty.html.

Distances were validated against Geoscience Australia calculations (http://www.ga.gov.au/geodesy/datums/vincenty_inverse.jsp).

Bearings were from multiple sources including http://williams.best.vwh.net/avform.htm#Crs.

References

Vincenty, T. 1975. Direct and Inverse Solutions of Geodesics on the Ellipsoid with application of Nested Equations. Survey Review, vol XXII no 176. http://www.ngs.noaa.gov/PUBS_LIB/inverse.pdf

See Also

destination

Examples

```
#get the distance of 1 degree longitude at each 5 degrees latitude from -90 to 90
distance(lat1=seq(-90,90,5),lon1=rep(0,37),lat2=seq(-90,90,5),lon2=rep(1,37),bearing=TRUE)
```

---

extract.data *Spatial Join of Points with Raster Grids*

Description

extract.data extracts data from raster object of class 'asc' (this and the adehabitat package), 'RasterLayer' (raster package) or 'SpatialGridDataFrame' (sp package) at specified locations. This represents a faster version of 'join.asc' of the adehabitat package that assumes all locations are within the map extents.

Note: there is no interpolation done here. The values reported are simply the values of the raster cell the point falls into.

Usage

```
## S3 method for class 'data'
extract(pts, x)
```

Arguments

pts	a two-column data frame or matrix with the x and y coordinates of the locations of interest.
x	a raster matrix of class 'asc' (this and the adehabitat package), 'RasterLayer' (raster package) or 'SpatialGridDataFrame' (sp package)

Details

Implements a faster version of 'join.asc' from the adehabitat package.

NOTE: this assumes all locations are within the extent of the raster map. Values outside the extent will be given a value of NA.

Value

Returns a vector equal in length to the number of locations in pnts.

Author(s)

Jeremy VanDerWal <jjvanderwal@gmail.com>

Examples

```
#create a simple object of class 'asc'
tasc = as.asc(matrix(1:50,nr=50,nc=50)); print(tasc)

#define some point locations
points = data.frame(x=runif(25,1,50),y=runif(25,1,50))

#extract the data
points$values = extract.data(points,tasc)

#show the data
print(points)
```

---

getXYcoords *Computes the X and Y Coordinates of the Pixels of a Raster Map*

Description

getXYcoords computes the geographical coordinates of the rows and columns of pixels of a raster map of class asc. Code & helpfile were modified from adehabitat package.

Usage

```
getXYcoords(w)
```

Arguments

w	an object of class asc.

Value

Returns a list with two components:

x	the x coordinates of the columns of pixels of the map
y	the y coordinates of the rows of pixels of the map

Author(s)

Jeremy VanDerWal <jjvanderwal@gmail.com>

Examples

```
tasc = as.asc(matrix(rep(x=1:10, times=1000),nr=100)); print(tasc)
getXYcoords(tasc)
```

---

grid.area *Create a Grid of Cell Areas or Perimeters*

Description

Creates a grid of cell areas or perimeters for spatial grids in geographic (lat-lon) projections.

Usage

```
grid.area(mat)

grid.perimeter(mat)
```

Arguments

mat	a matrix representing a raster of class 'asc' (this & adehabitat package), 'RasterLayer' (raster package) or 'SpatialGridDataFrame' (sp package)

Value

grid.area	Returns an ascii grid file which contains the values of the area in each cell.
grid.perimter	Returns an ascii grid file which contains the values of the perimeter in each cell.

Author(s)

Jeremy VanDerWal <jjvanderwal@gmail.com> & Lorena Falconi <lorefalconi@gmail.com>

Examples

```
#Create an ascii file
y=seq(10,50,0.5)
x=seq(140,180,0.5)
cellsize=0.5
data1=sample(160,140)
out1.asc=as.asc(matrix(data1,nc=y, nr=x), xll=min(x), yll=min(y), cellsize=cellsize)

grid.area(out1.asc)[,]

grid.perimeter(out1.asc)[,]
```

---

grid.info *Grid Information from Geographic (lat lon) Projections*

Description

Since spatial grids in geographic projections do not have equal area or perimeters, `grid.info` extracts perimeter & area related information for latitudinal bands with differing longitudinal widths.

Outputs lengths are in m using Vincenty's equation (`distance`)and areas in m2. Surface areas are calculated summing surface areas of spherical polygons as estimated using l'Huiller's formula.

Usage

```
grid.info(lats, cellsize, r = 6378137)
```

Arguments

`lats`	is a vector of latitudes representing the midpoint of grid cells
`cellsize`	is a single value (assuming square cells) or a two value vector (rectangular cells) representing the height (latitude) and width (longitude) of the cells
`r`	is a single value representing the radius of the globe in m. Default is for the WGS84 elipsoid

Value

a data.frame listing:

`lat`	the latitude representing the midpoint of the cell
`top`	length of the top of the cell (m)
`bottom`	length of the bottom of the cell (m)
`side`	length of the side of the cell (m)
`diagnal`	length of the diagnals of the cell (m)
`area`	area of the cell (m2)

Author(s)

Jeremy VanDerWal <jjvanderwal@gmail.com>

References

information on l'Huiller's formula http://williams.best.vwh.net/avform.htmformoreinfo) code for estimating area of polygon on sphere was modified from http://forum.worldwindcentral.com/showthread.php?t=20724

Examples

```
#show output for latitudes from -87.5 to 87.5 at 5 degree intervals
grid.info(lats=seq(-87.5,87.5,5), 5)
```

Istat *I Similarity Statistic for Quantifying Niche Overlap*

Description

Istat computes the I similarity statistic of Warren et al. 2008. It is a method for defining niche overlap from predictions of species' distributions.

NOTE: it is assumed the input data are of the same extent and cellsize, and all values are positive.

Usage

```
Istat(x, y, old = FALSE)
```

Arguments

x	a vector or matrix of data; the matrix can be a raster of class 'asc' (adehabitat package), 'RasterLayer' (raster package) or 'SpatialGridDataFrame' (sp package)
y	a vector or matrix of data with the same dimensions and class of 'x'
old	a boolean identifying if "old" equation is to be used (see description). This was kept for legacy issues.

Details

The I similarity statistic sums the pair-wise differences between two predictions to create a single value representing the similarity of the two distributions. The I similarity statistic ranges from a value of 0, where two distributions have no overlap, to 1 where two distributions are identical (Warren et al., 2008).

NOTE: updated to correct equation but not to worry about old... see explanation at http://enmtools.blogspot.com.au/2010_09_01_archive.html.

Value

A single value that is the I similarity statistic

Author(s)

Jeremy VanDerWal <jjvanderwal@gmail.com>

References

Warren, D. L., R. E. Glor, M. Turelli, and D. Funk. 2008. Environmental Niche Equivalency versus Conservatism: Quantitative Approaches to Niche Evolution. Evolution 62:2868-2883.

Examples

```
#create some simple objects of class 'asc'
tasc = as.asc(matrix(1:50,nr=50,nc=50)); print(tasc)
#modify the asc objects so that they are slightly different
tasc1 = tasc + runif(n = 2500, min = -1, max = 1)
tasc2 = tasc + rnorm(n = 2500, mean = 1, sd = 1)

#ensure all data is positive
tasc1 = abs(tasc1)
tasc2 = abs(tasc2)

#calculate the I similarity statistic
I = Istat(tasc1,tasc2)
print(I) #high niche overlap

#using a more variable map
tasc2 = tasc + rnorm(n = 2500, mean = 25, sd = 15);tasc2 = abs(tasc2)
I = Istat(tasc1,tasc2)
print(I) #lower niche overlap
```

Kappa *Kappa Statistic*

Description

Kappa estimates the Kappa statistic for model accuracy.

Usage

```
Kappa(mat)
```

Arguments

mat	a confusion matrix of class 'confusion.matrix' from confusion.matrix

Value

Returns a single value represting the Kappa statistic.

Author(s)

Jeremy VanDerWal <jjvanderwal@gmail.com>

See Also

auc, omission, sensitivity, specificity, prop.correct, confusion.matrix, accuracy

Examples

```
#create some data
obs = c(sample(c(0,1),20,replace=TRUE),NA); obs = obs[order(obs)]
pred = runif(length(obs),0,1); pred = pred[order(pred)]

#calculate the confusion matrix
mat = confusion.matrix(obs,pred,threshold=0.5)

#calculate the Kappa statistic
Kappa(mat)
```

---

lcmw *Least Cost Moving Windows Calculation*

Description

This is a moving window that for each cell returns the minimum 'cost' based on surrounding data cells and some dispersal distance cost.

Usage

```
lcmw(mat, mw, mnc)
```

Arguments

mat	a matrix of values that can be based on a raster dataset. Lower values should represent lower cost. The matrix can be a raster of class 'asc' (adehabitat package), 'RasterLayer' (raster package) or 'SpatialGridDataFrame' (sp package)
mw	a distance-cost matrix to be applied to each cell of 'mat'. This matrix can be dispersal costs. Lower values should represent lower cost.
mnc	an integer value representing the radius for 'mw' in number of cells.

Details

This method moves over the matrix of values, summing the moving window cost mw and the matrix mat, returning the minimum cost value. This was created to estimate the least cost path through time for all cells in a matrix (see example).

Value

A matrix of values of the same dimensions and class as input `mat`

Author(s)

Jeremy VanDerWal <jjvanderwal@gmail.com>

Examples

```
#create a simple object of class 'asc'
tasc = as.asc(matrix(1:100,nr=10,nc=10)); print(tasc)

#show the input matrix
print(tasc[1:10,1:10])

#vary the moving windows

###no cost window of 2 cell radius
tcost = matrix(0,nr=5,nc=5); print(tcost)
out = lcmw(tasc, tcost, 2); print(out[1:10,1:10])

###no cost with a circular radius of 2
tcost = matrix(NA,nr=5,nc=5)
#populate the distances
for (y in 1:5){
    for (x in 1:5){
        tcost[y,x] = sqrt((3-y)^2 + (3-x)^2)
    }
}

#remove distance values > max.num.cells
tcost[which(tcost>2)]=NA

#no cost matrix
tcost1 = tcost; tcost1[is.finite(tcost1)]=1; print(tcost1)
out = lcmw(tasc, tcost1, 2); print(out[1:10,1:10])

#linear cost
tcost = tcost/2; print(tcost)
out = lcmw(tasc, tcost, 2); print(out[1:10,1:10])
```

---

legend.gradient *Legend Gradient*

Description

`legend.gradient` creates and displays a gradient legend on a plot or image file. The place and size of the legend is defined by coordinates, previously identified.

Usage

```
legend.gradient(pnts, cols = heat.colors(100), limits = c(0, 1),
  title = "Legend", ...)
```

Arguments

pnts	x and y coordinates of the gradient location in the plot
cols	a set of 2 or more colors used in the image, to create the gradient
limits	to label the min and max values of the gradient in the legend
title	to specify the title of the legend
...	other graphical parameters defined by image() or plot()

Value

nothing is returned, a gradient legend is added to a plot or a image.

Author(s)

Lorena Falconi <lorefalconi@gmail.com>

Examples

```
#define a simple binary matrix
tmat = { matrix(c( 0,0,0,1,0,0,1,1,0,1,
                   0,0,1,0,1,0,0,0,0,0,
                   0,1,NA,1,0,1,0,0,0,1,
                   1,0,1,1,1,0,1,0,0,1,
                   0,1,0,1,0,1,0,0,0,1,
                   0,0,1,0,1,0,0,1,1,0,
                   1,0,0,1,0,0,1,0,0,0,
                   0,1,0,0,0,1,0,NA,NA,NA,
                   0,0,1,1,1,0,0,NA,NA,NA,
                   1,1,1,0,0,0,0,NA,NA,NA),nr=10,byrow=TRUE) }

#do the connected component labeling
tasc = ConnCompLabel(tmat)

# Create a color ramp
colormap=c("grey","yellow","yellowgreen","olivedrab1","lightblue4")

#create an image
image(tasc,col=colormap, axes=FALSE, xlab="", ylab="", ann=FALSE)

#points for the gradient legend
pnts = cbind(x =c(0.8,0.9,0.9,0.8), y =c(1.0,1.0,0.8,0.8))

#create the gradient legend
legend.gradient(pnts,colormap,c("Low","High"))
```

omission *Measures of Accuracy*

Description

Estimates different measures of accurracy given a confusion matrix.

Usage

```
omission(mat)

sensitivity(mat)

specificity(mat)

prop.correct(mat)
```

Arguments

mat	a confusion matrix of class 'confusion.matrix' from confusion.matrix

Value

returns single values representing the:

ommission	the ommission rate as a proportion of true occurrences misidentified given the defined threshold value
sensitivity	the sensitivity given the defined threshold value
specificity	the specificity given the defined threshold value
prop.correct	the proportion of the presence and absence records correctly identified given the defined threshold value

Author(s)

Jeremy VanDerWal <jjvanderwal@gmail.com>

See Also

auc, Kappa, confusion.matrix, accuracy

Examples

```
#create some data
obs = c(sample(c(0,1),20,replace=TRUE),NA); obs = obs[order(obs)]
pred = runif(length(obs),0,1); pred = pred[order(pred)]

#calculate the confusion matrix
mat = confusion.matrix(obs,pred,threshold=0.5)
```

```
#calculate the accuracy measures
omission(mat)
sensitivity(mat)
specificity(mat)
prop.correct(mat)
```

`optim.thresh` *Estimation of Optimal Threshold Values*

Description

`optim.thresh` estimates optimal threshold values given eight methods.

Note: this method will exclude any missing data.

Usage

```
optim.thresh(obs, pred, threshold = 101)
```

Arguments

`obs`	a vector of observed values which must be 0 for absences and 1 for occurrences
`pred`	a vector of the same length as `obs` representing the predicted values. Values must be between 0 & 1 representing a likelihood.
`threshold`	a single integer value representing the number of equal interval threshold values between 0 & 1

Value

Returns a list of the optimal thresholds for the different methods. If the list item is a single value, that is the optimal threshold but if two values are reported for the method, this represents the range in thresholds that are equal for that threshold selection method.

The returned list includes the single or range in thresholds selected using the following methods:

`min.occurence.prediction`
: is the minimum prediction for the occurrence (presence) records

`mean.occurence.prediction`
: is the mean prediction for the occurrence (presence) records

`'10.percent.omission'`
: is the threshold value or range in values that excludes approx. 10 percent of the occurrence records

`'sensitivity=specificity'`
: is the threshold value or range in values where sensitivity is equal to sensitivity

`'max.sensitivity+specificity'`
: is the threshold value or range in values that maximizes sensitivity plus specificity

maxKappa is the threshold value or range in values with the maximum Kappa statistic

max.prop.correct
: is the threshold value or range in values with the maximum proportion of presence and absence records correctly identified

min.ROC.plot.distance
: is the threshold value or range in values where the ROC curve is closest to point (0,1) (or perfect fit)

Author(s)

Jeremy VanDerWal <jjvanderwal@gmail.com>

See Also

accuracy, auc, Kappa, omission, sensitivity, specificity, prop.correct, confusion.matrix

Examples

```
#create some data
obs = c(sample(c(0,1),20,replace=TRUE),NA); obs = obs[order(obs)]
pred = runif(length(obs),0,1); pred = pred[order(pred)]

#calculate the optimal thresholds
optim.thresh(obs,pred)
```

PatchStat *Landscape Patch Statistics*

Description

PatchStat calculates the patch statistics for individual patches identified in a matrix of data. The matrix can be a raster of class 'asc' (adehabitat package), 'RasterLayer' (raster package) or 'SpatialGridDataFrame' (sp package).

Usage

```
PatchStat(mat, cellsize = 1, latlon = FALSE)
```

Arguments

mat
: a matrix of data with individual patches identified as with ConnCompLabel; The matrix can be a raster of class 'asc' (this & adehabitat package), 'RasterLayer' (raster package) or 'SpatialGridDataFrame' (sp package)

cellsize
: cell size (in meters) is a single value representing the width/height of cell edges (assuming square cells)

latlon
: boolean value representing if the data is geographic. If latlon == TRUE, matrix must be of class 'asc', 'RasterLayer' or 'SpatialGridDataFrame'

Details

The patch statistics are based on statistics calculated by fragstats http://www.umass.edu/landeco/research/fragstats/fragstats.html.

Value

a data.frame listing

patchID	the unique ID for each patch.
n.cell	the number of cells for each patch, specified in square meters.
n.core.cell	the number of cells in the core area, without the edge area.
n.edges.perimeter	the number of outer perimeter cell edges of the patch.
n.edges.internal	the number of internal cell edges of the patch.
area	the area of each patch comprising a landscape mosaic.
core.area	represents the interior area of the patch, greater than the specified depth-of-edge distance from the perimeter.
perimeter	the perimeter of the patch, including any internal holes in the patch, specified in meters.
perim.area.ratio	the ratio of the patch perimeter (m) to area (m2).
shape.index	the shape complexity, sum of each patches perimeter divided by the square root of patch area.
frac.dim.index	fractal dimension index reflects shape complexity across a range of spatial scales; approaches 2 times the logarithm of patch perimeter (m) divided by the logarithm of patch area (m2).
core.area.index	quantifies core area as a percentage of patch area.

Author(s)

Jeremy VanDerWal <jjvanderwal@gmail.com>

References

McGarigal, K., S. A. Cushman, M. C. Neel, and E. Ene. 2002. FRAGSTATS: Spatial Pattern Analysis Program for Categorical Maps. Computer software program produced by the authors at the University of Massachusetts, Amherst. Available at the following web site: www.umass.edu/landeco/research/fragstats/fragstats.html

See Also

ClassStat, ConnCompLabel

Examples

```
#define a simple binary matrix
tmat = { matrix(c( 0,0,0,1,0,0,1,1,0,1,
                   0,0,1,0,1,0,0,0,0,0,
                   0,1,NA,1,0,1,0,0,0,1,
                   1,0,1,1,1,0,1,0,0,1,
                   0,1,0,1,0,1,0,0,0,1,
                   0,0,1,0,1,0,0,1,1,0,
                   1,0,0,1,0,0,1,0,0,1,
                   0,1,0,0,0,1,0,0,0,1,
                   0,0,1,1,1,0,0,0,0,1,
                   1,1,1,0,0,0,0,0,0,1),nr=10,byrow=TRUE) }

#do the connected component labelling
ccl.mat = ConnCompLabel(tmat)
ccl.mat
image(t(ccl.mat[10:1,]),col=c('grey',rainbow(length(unique(ccl.mat))-1)))

#calculate the patch statistics
ps.data = PatchStat(ccl.mat)
ps.data
```

---

pnt.in.poly *Point in Polygon*

Description

pnt.in.poly works out if 2D points lie within the boundaries of a defined polygon.

Note: Points that lie on the boundaries of the polygon or vertices are assumed to be within the polygon.

Usage

```
pnt.in.poly(pnts, poly.pnts)
```

Arguments

pnts	a 2-column matrix or dataframe defining locations of the points of interest
poly.pnts	a 2-column matrix or dataframe defining the locations of vertices of the polygon of interest

Details

The algorithm implements a sum of the angles made between the test point and each pair of points making up the polygon. The point is interior if the sum is 2pi, otherwise, the point is exterior if the sum is 0. This works for simple and complex polygons (with holes) given that the hole is defined with a path made up of edges into and out of the hole.

This sum of angles is not able to consistently assign points that fall on vertices or on the boundary of the polygon. The algorithm defined here assumes that points falling on a boundary or polygon vertex are part of the polygon.

Value

A 3-column dataframe where the first 2 columns are the original locations of the points. The third column (names pip) is a vector of binary values where 0 represents points not with the polygon and 1 within the polygon.

Author(s)

Jeremy VanDerWal <jjvanderwal@gmail.com>

Examples

```
#define the points and polygon
pnts = expand.grid(x=seq(1,6,0.1),y=seq(1,6,0.1))
polypnts = cbind(x=c(2,3,3.5,3.5,3,4,5,4,5,5,4,3,3,3,2,2,1,1,1,1,2),
    y=c(1,2,2.5,2,2,1,2,3,4,5,4,5,4,3,3,4,5,4,3,2,2))

#plot the polygon and all points to be checked
plot(rbind(polypnts, pnts))
polygon(polypnts,col='#99999990')

#create check which points fall within the polygon
out = pnt.in.poly(pnts,polypnts)
head(out)

#identify points not in the polygon with an X
points(out[which(out$pip==0),1:2],pch='X')
```

---

put.data *Spatial Join of Points with Raster Grids - replace data*

Description

put.data replaces data in raster object of class 'asc' (this and adehabitat package) at specified locations.

Note: there is no interpolation done here. The values given replace the values of the raster cell the point falls into.

Usage

```
put.data(pts, x)
```

Arguments

pts	a three-column data frame or matrix with the x and y coordinates of the locations of interest and the third column being the z values to put in the ascii grid file.
x	a raster matrix of class 'asc' (this and the adehabitat package)

Details

Implements a faster version of 'join.asc' from the adehabitat package.

NOTE: this assumes all locations are within the extent of the raster map. Values outside the extent will be given a value of NA.

Value

Returns a raster matrix of class 'asc' equal in size to input 'x'.

Author(s)

Jeremy VanDerWal <jjvanderwal@gmail.com>

Examples

```
#create a simple object of class 'asc'
tasc = as.asc(matrix(1:50,nr=50,nc=50)); print(tasc)
## Not run: image(tasc)

#define some point locations
points = data.frame(x=runif(25,1,50),y=runif(25,1,50),z=50)

#put the new data
tasc = put.data(points,tasc)

#show the data
## Not run: image(tasc)
```

---

quick.map *Quick Map*

Description

quick.map creates and displays an image, identifying the threshold as the background color, and create the gradient legend in the map.

Usage

```
quick.map(sdm.asc, threshold, bkgd.col = "grey", cols = heat.colors(100),
  zlim = NULL, pnts = NULL, ...)
```

Arguments

sdm.asc	an object of class 'asc' (adehabitat package), 'RasterLayer' (raster package) or 'SpatialGridDataFrame' (sp package)
threshold	to indicate the threshold limit of sdm.asc
bkgd.col	to specify the background color
cols	a set of 2 or more colors to be used in the image and the gradient legend
zlim	to specify the upper an lower limits, which are going to be the labels of the gradient legend
pnts	location information for adding the legend.gradient
...	other graphical parameters defined by image() or plot()

Details

An image is created of the map requested. A gradient legend (legend.gradient) will be added if pnts (the position of the legend) is specified.

Value

Nothing is returned, an image is created.

Author(s)

Lorena Falconi <lorefalconi@gmail.com>

Examples

```
#create a matrix
tmat = { matrix(c( 0,0,0,1,0,0,1,1,0,1,
                   0,0,1,0,1,0,0,0,0,0,
                   0,1,NA,1,0,1,0,0,0,1,
                   1,0,1,1,1,0,1,0,0,1,
                   0,1,0,1,0,1,0,0,0,1,
                   0,0,1,0,1,0,0,1,1,0,
                   1,0,0,1,0,0,1,0,0,0,
                   0,1,0,0,0,1,0,0,0,1,
                   0,0,1,1,1,0,0,1,1,1,
                   1,1,1,0,0,0,0,1,1,1),nr=10,byrow=TRUE) }

#do the connected component labeling
tasc = ConnCompLabel(tmat)

#put in the gradient scale
pnts = cbind(x =c(1.1,1.2,1.2,1.1), y =c(0.9,0.9,0.7,0.7))

# Set the map and gradient leyend colors
tasc.col=colorRampPalette(c("yellow","orange", "red"))(5)

#Create an image with the gradient legend
quick.map(tasc,0.09,bkgd.col = 'darkgrey', cols=tasc.col,
```

```
    axes=FALSE, xlim=c(0.0,1.35))

########################
# Create an image with two colors: below the threshold and
# above the threshold

# The next version of SDM Tools will let you create the legend.gradient
# at a specific side of your image, and the user would not need to set
# the coordinates.

# To create the legend.gradient at the bottom left of your image without
# setting up the coordinates at the image you can do this:

xlim = c(-0.5,1)
ylim = c(0,1)
wid = diff(xlim)*0.05
ht = diff(ylim)*0.1
xvals = c(xlim[1]+wid,xlim[1]+2*wid,xlim[1]+2*wid,xlim[1]+wid)
yvals = c(ylim[1]+ht,ylim[1]+ht,ylim[1]+2*ht,ylim[1]+2*ht)

#Create the points for the legend.gradient
pnts=(cbind(xvals,yvals))

# Set the images colors: above the threshold is black and
# below the threshold is darkgrey.
quick.map(tasc,0.09,bkgd.col = 'darkgrey', cols="black",
    axes=FALSE, xlim=c(-0.8, 1))
```

`read.asc` *ESRI ASCII Raster File Import And Export*

Description

`read.asc` and `read.asc.gz` reads ESRI ArcInfo ASCII raster file either uncompressed or compressed using gzip.

`write.asc` and `write.asc.gz` writes an asc object to a ESRI ArcInfo ASCII raster file. The output can be either compressed or uncompressed.

These functions are faster methods based on the adehabitat import.asc and export.asc.

`write.asc2` and `write.asc2.gz` are even faster implementations but have less error checking.

`image.asc` and `print.asc` are generic methods associated with plotting & summarizing data of class 'asc'; they were modified from adehabitat package.

Usage

```
read.asc(file, gz = FALSE)
```

```
read.asc.gz(file)

write.asc(x, file, gz = FALSE)

write.asc.gz(x, file)

write.asc2(x, file, sigdig = 0, gz = FALSE)

write.asc2.gz(x, file, sigdig = 0)

## S3 method for class 'asc'
image(x, col = gray((240:1)/256), clfac = NULL, ...)

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

Arguments

file	a character string representing the filename of the input/output file. The file extension should always be '.asc'.
gz	defines if the object is or should be compressed using gzip
x	an object of class 'asc' as defined in the adehabitat package
sigdig	is the number of significant digits to write when creating the ascii grid file
col	for maps of type "numeric", the colors to be used (see help(par))
clfac	for maps of type "factor", a character vector giving the names of colors for each level of the factor (see help(colasc))
...	additional arguments to be passed to the generic function image or print

Details

Implements a faster version of import.asc or export.asc from the adehabitat package. In addition, files can be read in and written to in gzip compressed format.

Generic methods of print and image were modified from adehabitat. Further details of them are found there.

Value

Returns a raster matrix of the class 'asc' defined in the adehabitat package with the following attributes:

xll	the x coordinate of the center of the lower left pixel of the map
yll	the y coordinate of the center of the lower left pixel of the map
cellsize	the size of a pixel on the studied map
type	either 'numeric' or 'factor'
levels	if type = 'factor', the levels of the factor.

Author(s)

Jeremy VanDerWal <jjvanderwal@gmail.com>

Examples

```
#create a simple object of class 'asc'
tasc = as.asc(matrix(rep(x=1:10, times=1000),nr=100)); print(tasc)

#write out the raster grid file
write.asc(tasc,'t.raster.asc')
write.asc.gz(tasc,'t.raster.asc') #actually save file name as t.raster.asc.gz

#read in the raster grid files
tasc2 = read.asc('t.raster.asc')
tasc3 = read.asc.gz('t.raster.asc.gz')

#remove the temporary raster
unlink(c('t.raster.asc','t.raster.asc.gz'))
```

Scalebar *Scalebar for Projected Maps*

Description

Scalebar adds a distance scalebar onto a projected map. It is not appropriate for geographic projections.

Usage

```
Scalebar(x, y, distance, unit = "km", scale = 1, t.cex = 0.8)
```

Arguments

x	the x-axis position for the lower left corner of the bar
y	the x-axis position for the lower left corner of the bar
distance	the distance for which the scale bar should represent
unit	the units to report as the scaling
scale	the scaling factor to rescale the distance to a different unit. e.g., if your map is in m and want the scalebar to be in km, use a scale of 0.01
t.cex	the scaling of the font size to be used for the scalebar

Value

nothing is returned, simply a scalebar is added to a plot.

Author(s)

Jeremy VanDerWal <jjvanderwal@gmail.com>

Examples

```
#create a simple object of class 'asc'
tasc = as.asc(matrix(1:50,nr=50,nc=50)); print(tasc)

#plot the image
image(tasc,axes=FALSE,ann=FALSE)

#add a distance scalebar
Scalebar(x=5,y=5,distance=20) #show values in km
Scalebar(x=5,y=10,distance=20,unit='m',scale=1000) #show values in meters
```

SigDiff *Identify Regions of Significant Differences*

Description

SigDiff computes the significance of the pairwise differences relative to the mean and variance of all differences between the two input datasets. This is useful for identifying regions of significant difference between two datasets (e.g., different DEMs (Januchowski et al. 2010) or different species distribution model predictions (Bateman et al 2010)).

ImageDiff is a wrapper to the image.asc command in adehabitat package that uses the result from SigDiff to create an image mapping the regions of significant differences (positive and negative).

NOTE: it is assumed the input data are of the same extent and cellsize.

Usage

```
SigDiff(x, y, pattern = TRUE)

ImageDiff(tasc, sig.levels = c(0.025, 0.975), tcol = terrain.colors(3), ...)
```

Arguments

x	a vector or matrix of data; the matrix can be of can be a raster of class 'asc' (adehabitat package), 'RasterLayer' (raster package) or 'SpatialGridDataFrame' (sp package)
y	a vector or matrix of data with the same dimensions and class of 'x'
pattern	logical value defining if differences are respective to relative patterning (TRUE) or absolute values (FALSE)
tasc	a matrix of probability values (0 to 1) likely created by SigDiff; The matrix can be a raster of class 'asc' (adehabitat package), 'RasterLayer' (raster package) or 'SpatialGridDataFrame' (sp package)

sig.levels	the significance levels to define significantly above and below. Default settings represent significance at the 0.05 level
tcol	a set of 3 colors for use in the image to represent significantly lower or greater, and not significant
...	other graphical parameters defined by image() or plot()

Value

SigDiff returns a vector or matrix of the same dimensions and class of the input representing the significance of the pairwise difference relative to the mean and variance of all differences between the two inputs.

ImageDiff returns nothing but creates an image of the areas of significant differences

Author(s)

Stephanie Januchowski <stephierenee@gmail.com>

References

Januchowski, S., Pressey, B., Vanderwal, J. & Edwards, A. (2010) Characterizing errors in topographic models and estimating the financial costs of accuracy. International Journal of Geographical Information Science, In Press.

Bateman, B.L., VanDerWal, J., Williams, S.E. & Johnson, C.N. (2010) Inclusion of biotic interactions in species distribution models improves predictions under climate change: the northern bettong Bettongia tropica, its food resources and a competitor. Journal of Biogeography, In Review.

Examples

```
#create some simple objects of class 'asc'
tasc = as.asc(matrix(1:50,nr=50,nc=50)); print(tasc)
#modify the asc objects so that they are slightly different
tasc1 = tasc + runif(n = 2500, min = -1, max = 1)
tasc2 = tasc + rnorm(n = 2500, mean = 1, sd = 1)

#create graphical representation
par(mfrow=c(2,2),mar=c(1,1,4,1))
image(tasc1,main='first grid',axes=FALSE)
image(tasc2,main='second grid',axes=FALSE)

#get significant difference by spatial patterning
out = SigDiff(tasc1,tasc2)
ImageDiff(out,main="Pattern Differences",axes=FALSE)

#get significant difference
out = SigDiff(tasc1,tasc2,pattern=FALSE)
ImageDiff(out,main="Absolute Differences",axes=FALSE)
legend('topleft',legend=c('-ve','ns','+ve'),title='significance',
  fill=terrain.colors(3),bg='white')
```

slope *Slope and aspect calculations*

Description

slope and aspect calculates the slope and aspect of raster surfaces of class 'asc' (SDMTools & adehabitat packages), 'RasterLayer' (raster package) or 'SpatialGridDataFrame' (sp package).

Methods are based on Burrough and McDonell (1998).

Usage

```
slope(mat, latlon = FALSE)

aspect(mat, latlon = FALSE)
```

Arguments

mat	a matrix of data representing z heights. Matrix can be a raster of class 'asc' (adehabitat package), 'RasterLayer' (raster package) or 'SpatialGridDataFrame' (sp package)
latlon	boolean value representing if the data is geographic.

Details

Slope returns values representing the 'rise over run' with "run" units representing cellsize if latlon=FALSE or km if latlon=TRUE. This can be changed to percentage (multiply by 100) or to degrees by ATAN (output) * 57.29578.

Aspect returns the direction (0 to 360) with North being 0. Values of -1 are flat areas with no slope or aspect.

As this method requires information from the surrounding cells, missing data (NAs or edges) are populated with the value from the 'cell-of-interest').

Value

an object of the same class as mat.

Author(s)

Jeremy VanDerWal <jjvanderwal@gmail.com>

References

Burrough, P. A. and McDonell, R.A., 1998. Principles of Geographical Information Systems (Oxford University Press, New York), p. 190.

Examples

```
#define a simple asc with some slope and direction
tasc = as.asc(matrix(1:50,nr=10,nc=5),yll=75); tasc[,]
slope(tasc)[,] #show the output of slope
aspect(tasc)[,] #show the output of the aspect

#define a FLAT simple asc
tasc = as.asc(matrix(10,nr=10,nc=5),yll=75); tasc[,]
slope(tasc)[,] #show the output of slope
aspect(tasc)[,] #show the output of the aspect
```

---

wt.mean *Weighted mean, variance and standard deviation calculations*

Description

wt.mean calculates the mean given a weighting of the values.

wt.var is the unbiased variance of the weighted mean calculation using equations of GNU Scentific Library (http://www.gnu.org/software/gsl/manual/html_node/Weighted-Samples.htmland.

wt.sd is the standard deviation of the weighted mean calculated as the sqrt of wt.var.

Note: NA data is automatically ommitted from analysis.

Usage

```
wt.mean(x, wt)

wt.var(x, wt)

wt.sd(x, wt)
```

Arguments

x	is a vector of numerical data.
wt	is a vector of equal length to x representing the weights.)

Value

returns a single value from analysis requested.

Author(s)

Jeremy VanDerWal <jjvanderwal@gmail.com>

Examples

```
#define simple data
x = 1:25 # set of numbers
wt = runif(25) #some arbitrary weights

#display means & variances (unweighted and then weighted)
mean(x); wt.mean(x,wt)
var(x); wt.var(x,wt)
sd(x); wt.sd(x,wt)
```

ZonalStat *Landscape Zonal Statistics*

Description

`ZonalStat` calculates the statistics of data for specified zones of two matrices of data. The matrix can be a raster of class 'asc' (adehabitat package), 'RasterLayer' (raster package) or 'SpatialGridDataFrame' (sp package).

Usage

```
ZonalStat(mat, zones, FUN = "all")
```

Arguments

`mat`	a matrix of data to be summarized; The matrix can be a raster of class 'asc' (adehabitat package), 'RasterLayer' (raster package) or 'SpatialGridDataFrame' (sp package)
`zones`	a matrix of data with individual patches identified as with `ConnCompLabel`; The matrix must be of the same size & extent as `mat`
`FUN`	a single or vector of functions to be applied to each 'zone'; the default of 'all' will calculate min, 1st quarter, median, 3rd quarter, max, mean, standard deviation and n

Details

The code summarizes the data for defined zones. Nearly any function can be used for summarizing the data.

The FUN defined with 'all' as one of or the only function will append the functions of min, 1st quarter, median, 3rd quarter, max, mean, standard deviation and n to what is being calculated.

Value

a data.frame listing

zone	the unique ID for each zone.
functions...	a column for each of the functions identified

The data.frame will have an atribute defining the number of NA values that were excluded from the analysis.

Author(s)

Jeremy VanDerWal <jjvanderwal@gmail.com>

Examples

```
#define a simple binary matrix
tmat = { matrix(c( 0,0,0,1,0,0,1,1,0,1,
                   0,0,1,0,1,0,0,0,0,0,
                   0,1,NA,1,0,1,0,0,0,1,
                   1,0,1,1,1,0,1,0,0,1,
                   0,1,0,1,0,1,0,0,0,1,
                   0,0,1,0,1,0,0,1,1,0,
                   1,0,0,1,0,0,1,0,0,1,
                   0,1,0,0,0,1,0,0,0,1,
                   0,0,1,1,1,0,0,0,0,1,
                   1,1,1,0,0,0,0,0,0,1),nr=10,byrow=TRUE) }

#do the connected component labelling
ccl.mat = ConnCompLabel(tmat)
ccl.mat #this is the zone matrix to be used

#create a random data matrix
data.mat = matrix(runif(100),nr=10,nc=10)
data.mat

#calculate the zonal statistics
zs.data = ZonalStat(data.mat,ccl.mat,FUN='all')
zs.data

#just calculate the sum
zs.data = ZonalStat(data.mat,ccl.mat,FUN='sum')
zs.data

#calculate sum & n & 'all' and show when a function is not defined
zs.data = ZonalStat(data.mat,ccl.mat,
    FUN=c('sum','length','not.a.function','all'))
zs.data
attr(zs.data,'excluded NAs') #show how many NAs were omitted from analysis
```

Index