Package ‘spatialEco’

November 17, 2023

Type Package

Title Spatial Analysis and Modelling Utilities

Version 2.0-2

Date 2023-11-17

Description Utilities to support spatial data manipulation, query, sampling and modelling in ecological applications. Functions include models for species population density, spatial smoothing, multivariate separability, point process model for creating pseudo- absences and sub-sampling, Quadrant-based sampling and analysis, auto-logistic modeling, sampling models, cluster optimization, statistical exploratory tools and raster-based metrics.

Depends R (>= 4.2)

Imports sf, terra

Suggests spatstat.geom (>= 3.0-3), spatstat.explore, spdep, ks, cluster, readr, RCurl, RANN, rms, yaImpute, mgcv, zyp, SpatialPack (>= 0.3), MASS, caret, dplyr, earth, Matrix, gstat, spatstat.data, methods, units, sp, stringr, lwgeom, geodata

Maintainer Jeffrey S. Evans `<jeffrey_evans@tnc.org>`

License GPL-3

URL `https://github.com/jeffreyevans/spatialEco`, `https://jeffreyevans.github.io/spatialEco/`

BugReports `https://github.com/jeffreyevans/spatialEco/issues`

NeedsCompilation no

Repository CRAN

LazyData true

Encoding UTF-8

RoxygenNote 7.2.3

Author Jeffrey S. Evans [aut, cre] (<`https://orcid.org/0000-0002-5533-7044`>),
Melanie A. Murphy [ctb],
Karthik Ram [ctb]

Date/Publication 2023-11-17 22:40:02 UTC

R topics documented:

`all_pairwise` *All pairwise combinations*

Description

Creates all pairwise combinations list for iteration

Usage

```
all_pairwise(x)
```

Arguments

`x`	A numeric or character vector

Details

This returns a list of vector combinations starting with pairwise, as the first nested list element, then in groups of threes, fours, to length of the vector.

Value

A list object with increasing all combination objects, the first list element are the pairwise comparisons

Author(s)

Jeffrey S. Evans <jeffrey_evans<at>tnc.org>

Examples

```
classes <- paste0("class", 1:10)

all_pairwise(classes)[[1]]

#### How to use as an iterator
# dataframe with 4 cols, 100 rows
d <- as.data.frame(matrix(runif(100*4), 100, 4))
  names(d) <- paste0("class", 1:4)

( idx <- all_pairwise(colnames(d))[[1]] )

opar <- par(no.readonly=TRUE)
  par(mfrow=c(2,3))
    lapply(idx, function(i) {
      plot(d[,i[1]], d[,i[2]], main=paste0(i[1], " vs ", i[2]) )
    })
par(opar)
```

`annulus.matrix` *Annulus matrix*

Description

Creates a square matrix representing annulus position values of 1 and defined null

Usage

```
annulus.matrix(scale = 3, inner.scale = 0, outer.scale = 0, null.value = 0)
```

Arguments

`scale`	Number of rings (defines dimensions of matrix)
`inner.scale`	Number of inner rings to set to null.value
`outer.scale`	Number of outer rings to set to null.value
`null.value`	Value to set inner and outer scale(s) to

Details

This function will return a matrix of 1 and defined null.value based on a specification of the scale, inner scale and outer scale. The scale defines how many rings will be represented in the matrix based on (2 * scale - 1). So, a scale of 3 will result in a 5x5 matrix. The inner.scale and outer.scale arguments represent the > and < rings that will be set to the defined null.value (see examples). The resulting matrix can be used as the specified window in a focal function.

Value

A matrix object with defined null.value and 1, representing retained rings

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
annulus.matrix(5)                  # 5 concentric rings
annulus.matrix(5, 3)               # 5 concentric rings with the 3 inner set to 0
annulus.matrix(5, 3, null.value=NA) # 5 concentric rings with the 3 inner set to NA
annulus.matrix(5, 3, 5)            # 5 rings with 3 inner and 5 outer set to 0
annulus.matrix(9, 3, 7)            # 9 rings with 3 inner and 7 outer set to 0
```

`ants` *Ant Biodiversity Data*

Description

Roth et al., (1994) Costa Rican ant diversity data

Format

A data.frame with 82 rows (species) and 5 columns (covertypes):

species	Ant species (family)
Primary.Forest	Primary forest type
Abandoned.cacao.plantations	Abandoned cacao plantations type
Productive.cacao.plantations	Active cacao plantations type
Banana.plantations	Active banana plantations type

References

Roth, D. S., I. Perfecto, and B. Rathcke (1994) The effects of management systems on ground-foraging ant diversity in Costa Rica. Ecological Applications 4(3):423-436.

`aspline.downscale` *Raster Downscale using adaptive regression splines*

Description

Downscales a raster to a higher resolution raster multivariate adaptive regression splines (MARS).

Usage

```
aspline.downscale(
  x,
  y,
  add.coords = TRUE,
  keep.model = FALSE,
  grid.search = FALSE,
  plot = FALSE,
  ...
)
```

Arguments

`x`	A terra SpatRaster object representing independent variable(s)
`y`	A terra SpatRaster object representing dependent variable
`add.coords`	(FALSE/TRUE) Add spatial coordinates to model
`keep.model`	(FALSE/TRUE) Keep MARS model (earth class object)
`grid.search`	(FALSE/TRUE) perform a hyper-parameter grid se
`plot`	(FALSE/TRUE) Plot results
`...`	Additional arguments passed to earth

Details

This function uses Multivariate Adaptive Regression Splines, to downscale a raster based on higher-resolution or more detailed raster data specified as covariate(s). This is similar to the raster.downsample function which uses a robust regression and is a frequentest model for fitting linear asymptotic relationships whereas, this approach is for fitting nonparametric functions and should be used when the distributional relationship are complex/nonlinear. Using add.coords adds spatial coordinates to the model, including creating the associated rasters for prediction.

Value

A list object containing:

- downscale Downscaled terra SpatRaster object
- GCV Generalized Cross Validation (GCV)
- GRSq Estimate of the predictive power
- RSS Residual sum-of-squares (RSS)
- RSq R-square
- model earth MARS model object (if keep.model = TRUE)

Author(s)

Jeffrey S. Evans jeffrey_evans@tnc.org

References

Friedman (1991) Multivariate Adaptive Regression Splines (with discussion) Annals of Statistics 19(1):1–141

Examples

```
if (require(geodata, quietly = TRUE)) {
library(terra)
library(geodata)

# Download example data (requires geodata package)
  elev <- elevation_30s(country="SWZ",  path=tempdir())
```

```
    slp <- terrain(elev, v="slope")
  x <- c(elev,slp)
        names(x) <- c("elev","slope")
  tmax <- worldclim_country(country="SWZ", var="tmax",
                              path=tempdir())
    tmax <- crop(tmax[[1]], ext(elev))
  names(tmax) <- "tmax"

tmax.ds <- aspline.downscale(x, tmax, add.coords=TRUE, keep.model=TRUE)
  plot(tmax.ds$model)

  # plot prediction and parameters
  opar <- par(no.readonly=TRUE)
    par(mfrow=c(2,2))
      plot(tmax, main="Original Temp max")
      plot(x[[1]], main="elevation")
      plot(x[[2]], main="slope")
      plot(tmax.ds$downscale, main="Downscaled Temp max")
  par(opar)

} else {
  cat("Please install geodata package to run example", "\n")
}
```

`background` *Background sample*

Description

Creates a point sample that can be used as a NULL for SDM's and other modeling approaches.

Usage

```
background(
  x,
  p = 1000,
  known = NULL,
  d = NULL,
  type = c("regular", "random", "hexagon", "nonaligned")
)
```

Arguments

`x`	A sf class polygon defining sample region
`p`	Size of sample
`known`	An sf POINT class of known locations with same CSR as x
`d`	Threshold distance for known proximity
`type`	Type of sample c("systematic", "random", "hexagon", "nonaligned")

Details

This function creates a background point sample based on an extent or polygon sampling region. The known argument can be used with d to remove sample points based on distance-based proximity to existing locations (eg., known species locations). The size (p) of the resulting sample will be dependent on the known locations and the influence of the distance threshold (d). As such, if the know and d arguments are provided the exact value provided in p will not be returned.

Value

A sf POINT feature class or data.frame with x,y coordinates

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
library(sf)

# define study area
sa <- suppressWarnings(st_cast(st_read(
        system.file("shape/nc.shp",
        package="sf")), "POLYGON"))
  sa <- sa[10,]

# create "known" locations
locs <- st_sample(sa, 50)
  st_crs(locs) <- st_crs(sa)

# systematic sample using extent polygon
e <- st_as_sf(st_as_sfc(st_bbox(sa)))
  st_crs(e) <- st_crs(sa)
s <- background(e, p=1000, known=locs, d=1000)
  plot(st_geometry(s), pch=20)
    plot(st_geometry(locs), pch=20, col="red", add=TRUE)

# systematic sample using irregular polygon
s <- background(sa, p=1000, known=locs, d=1000)
  plot(st_geometry(sa))
    plot(st_geometry(s), pch=20, add=TRUE)
      plot(st_geometry(locs), pch=20, col="red", add=TRUE)

# random sample using irregular polygon
s <- background(sa, p=500, known=locs,
                d=1000, type="random")
  plot(st_geometry(sa))
    plot(st_geometry(s), pch=20, add=TRUE)
      plot(st_geometry(locs), pch=20, col="red", add=TRUE)
```

`bbox_poly` *Bounding box polygon*

Description

Creates a polygon from a vector or raster extent

Usage

```
bbox_poly(x)
```

Arguments

`x`	An sf or terra object or vector of bounding coordinates

Details

If not a spatial object, expected order of input for x is: xmin, ymin, xmax, ymax. Where; xmin, ymin and the coordinates of top left corner of the bounding box and xmax, ymax represent the bottom right corner. The maximum value of xmax is width of the extent while maximum value of ymax is the height of the extent.

Value

A single feature sf class polygon object

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
if(require(sp, quietly = TRUE)) {
library(terra)
library(sf)
  data(meuse, package = "sp")
  meuse <- st_as_sf(meuse, coords = c("x", "y"), crs = 28992,
                    agr = "constant")

# raster (terra)
r <- rast(ext(meuse))
  r[] <- runif(ncell(r))
 crs(r) <- "epsg:28992"
e <- bbox_poly(r)

plot(r)
  plot(st_geometry(e), border="red", add=TRUE)

# extent vector
```

```
e <- bbox_poly(c(178605, 329714, 181390, 333611))
  plot(e)

# vector bounding box
e <- bbox_poly(meuse)

plot(st_geometry(meuse), pch=20)
  plot(st_geometry(e), add=TRUE)

} else {
  cat("Please install sp package to run this example", "\n")
}
```

`bearing.distance` *Bearing and Distance*

Description

Calculates a new point [X,Y] based on defined bearing and distance

Usage

```
bearing.distance(x, y, distance, azimuth, EastOfNorth = TRUE)
```

Arguments

`x`	x coordinate
`y`	y coordinate
`distance`	Distance to new point (in same units as x,y)
`azimuth`	Azimuth to new point
`EastOfNorth`	Specified surveying convention

Details

East of north is a surveying convention and defaults to true.

Value

a new point representing location of baring and distance

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
pt <- cbind( x=480933, y=4479433)
bearing.distance(pt[1], pt[2], 1000, 40)
```

`breeding.density` *Breeding density areas (aka, core habitat areas)*

Description

Calculates breeding density areas base on population counts and spatial point density.

Usage

```
breeding.density(x, pop, p = 0.75, bw = 6400, b = 8500, self = TRUE)
```

Arguments

`x`	sf POINT object
`pop`	Population count/density column in x
`p`	Target percent of population
`bw`	Bandwidth distance for the kernel estimate (default 8500)
`b`	Buffer distance (default 8500)
`self`	(TRUE/FALSE) Should source observations be included in density (default TRUE)

Details

The breeding density areas model identifies the Nth-percent population exhibiting the highest spatial density and counts/frequency. It then buffers these points by a specified distance to produce breeding area polygons. If you would like to recreate the results in Doherty et al., (2010), then define bw = 6400m and b[if p < 0.75 b = 6400m, | p >= 0.75 b = 8500m]

Value

A list object with:

- pop.pts sf POINT object with points identified within the specified p
- pop.area sf POLYGON object of buffered points specified by parameter b
- bandwidth Specified distance bandwidth used in identifying neighbor counts
- buffer Specified buffer distance used in buffering points for pop.area
- p Specified population percent

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

References

Doherty, K.E., J.D. Tack, J.S. Evans, D.E. Naugle (2010) Mapping breeding densities of greater sage-grouse: A tool for range-wide conservation planning. Bureau of Land Management. Number L10PG00911

Examples

```
library(sf)

n=1500
bb <- rbind(c(-1281299,-761876.5),c(1915337,2566433.5))
bb.mat <- round(cbind(c(bb[1,1], bb[1,2], bb[1,2], bb[1,1]),
                  c(bb[2,1], bb[2,1], bb[2,2], bb[2,2])),2)
 bbp <- st_sfc(st_polygon(list(rbind(bb.mat, bb.mat[1,]))))
   pop <- st_as_sf(st_sample(bbp, n, type = "random"))
  st_geometry(pop) <- "geometry"
     pop$ID <- 1:nrow(pop)
  pop$counts <- round(runif(nrow(pop), 1,250),0)

    bd75 <- breeding.density(pop, pop='counts', p=0.75, b=8500, bw=6400)
      plot(st_geometry(bd75$pop.area), border = NA,
        main='75% breeding density areas', col="grey")
         plot(st_geometry(pop), pch=20, col='black', add=TRUE)
         plot(st_geometry(bd75$pop.pts), pch=20, col='red', add=TRUE)
      legend("bottomright", legend=c("selected areas","selected sites", "all sites"),
             bg="white", fill=c("grey","red", "black"), pt.cex = 2)
```

`built.index` *built index*

Description

Remote sensing built-up index

Usage

```
built.index(
  green,
  red,
  nir,
  swir1,
  swir2,
  L = 0.5,
  method = c("Bouhennache", "Zha", "Xu")
)
```

Arguments

`green`	Green band (0.53 - 0.59mm), landsat 5&7 band 3, OLI (landsat 8) band 3
`red`	Red band (0.636 - 0.673mm), landsat 5&7 band 3, OLI (landsat 8) band 4
`nir`	Near infrared band (0.851 - 0.879mm) landsat 5&7 band 4, OLI (landsat 8) band 5

`swir1`	short-wave infrared band 1 (1.566 - 1.651mm), landsat 5&7 band 5, OLI (landsat 8) band 6
`swir2`	short-wave infrared band 2 (2.11 - 2.29mm), landsat 5&7 band 7, OLI (landsat 8) band 7
`L`	The L factor for the savi index
`method`	Method to use for index options are "Bouhennache", "Zha", "Xu"

Details

This function calculates the built-up index. Three methods are available:

- Bouhennache is a new method that uses a larger portion of the VIR/NIR following OLI bands (((b3+b4+b7)-b6)/3) / (((b3+b4+b7)+b6)/3)
- Zha is the original band ratio method using TM5 ndbi = (b5 - b4) / (b5 + b4)
- Xu is a modification to eliminate noise using ETM+7 (ndbi-((savi-nndwi)/2) / (ndbi+((savi-nndwi)/2)

Generally water has the highest values where built-up areas will occur in the mid portion of the distribution. Since Bouhennache et al (2018) index exploits a larger portion of the visible (Vis) and infra red spectrum, vegetation will occur as the lowest values and barren will exhibit greater values than the vegetation and lower values than the built-up areas.

Band wavelength (nanometers) designations for landsat TM4, TM5 and ETM+7

- band-2 0.52-0.60 (green)
- band-3 0.63-0.69 (red)
- band-4 0.76-0.90 (NIR)
- band-5 1.55-1.75 (SWIR 1)
- band-7 2.09-2.35 (SWIR 2)

OLI (Landsat 8)

- band-3 0.53-0.59 (green)
- band-4 0.64-0.67 (red)
- band-5 0.85-0.88 (NIR)
- band-6 1.57-1.65 (SWIR 1)
- band-7 2.11-2.29 (SWIR 2)

Value

A terra raster object of the built index

Author(s)

Jeffrey S. Evans jeffrey_evans@tnc.org

References

Bouhennache, R., T. Bouden, A. Taleb-Ahmed & A. Chaddad(2018) A new spectral index for the extraction of built-up land features from Landsat 8 satellite imagery, Geocarto International 34(14):1531-1551

Xu H. (2008) A new index for delineating built-up land features in satellite imagery. International Journal Remote Sensing 29(14):4269-4276.

Zha G.Y., J. Gao, & S. Ni (2003) Use of normalized difference built-up index in automatically mapping urban areas from TM imagery. International Journal of Remote Sensing 24(3):583-594

Examples

```
library(terra)
 lsat <- rast(system.file("/extdata/Landsat_TM5.tif", package="spatialEco"))
   plotRGB(lsat, r=3, g=2, b=1, scale=1.0, stretch="lin")

 # Using Bouhennache et al., (2018) method (needs green, red, swir1 and swir2)
 ( bouh <- built.index(red = lsat[[3]], green = lsat[[2]], swir1 = lsat[[5]],
                       swir2 = lsat[[6]]) )
    plotRGB(lsat, r=3, g=2, b=1, scale=1, stretch="lin")
      plot(bouh, legend=FALSE, col=rev(terrain.colors(100, alpha=0.35)),
       add=TRUE )

 # Using simple Zha et al., (2003) method (needs nir and swir1)
 ( zha <- built.index(nir = lsat[[4]], swir1 = lsat[[5]], method = "Zha") )
   plotRGB(lsat, r=3, g=2, b=1, scale=1, stretch="lin")
     plot(zha, legend=FALSE, col=rev(terrain.colors(100, alpha=0.35)), add=TRUE )

 # Using Xu (2008) normalized modification of Zha (needs green, red, nir and swir1)
 ( xu <- built.index(green= lsat[[2]], red = lsat[[3]], nir = lsat[[4]],
                      swir1 = lsat[[5]], , method = "Xu") )
   plotRGB(lsat, r=3, g=2, b=1, scale=1, stretch="lin")
     plot(xu, legend=FALSE, col=rev(terrain.colors(100, alpha=0.35)), add=TRUE )
```

`cgls_urls` *Provide URL's for Copernicus Global Land Service datasets*

Description

Returns URL's of a product/version/resolution

Usage

```
cgls_urls(
  dates = NULL,
  resolution = c(1000, 300),
```

```
    product = c("fapar", "fcover", "lai", "ndvi"),
    ver = c("newest", "v1", "v2", "v3")
  )
```

Arguments

`dates`	Dates to subset default is NULL, returns all products
`resolution`	The product resolution c("1km", "300m"),
`product`	Which product to query options are "fapar",
`ver`	Product version options are "newest", "v1", "v2", "v3"

Details

Provides a query of the ESA's Copernicus Global Land Service global The query is performed on the manifest files and return URL's however, to download data you will need login credentials which, can be acquired from: http://land.copernicus.eu

If provided, dates need to be in a "YYYY-MM-DD" format. The dates are an explicit search string and can contain dates that are not in the imagery. As such, the user should generate a daily date string representing the range of the desired download as not to have to guess the available dates. Also note that multiple processing versions of a given image are retained in the manifest. This means that if you download a previous processing version, it could be an invalid image. It is highly recommended that you do not change the default ver="newest" argument unless there is a specific reason to.

Available products

- fapar Fraction of photosynthetically active radiation absorbed by the vegetation
- fcover Fraction of green vegetation cover
- lai Leaf Area index
- ndvi Normalized Difference Vegetation Index

Not yet implemented; Soil Water Index, Surface Soil Moisture, and Land Surface Temperature.

Value

A vector of download URL's for the products

Author(s)

Jeffrey S. Evans jeffrey_evans@tnc.org

Examples

```
# Create date string for query
d <- seq(as.Date("2020/05/01"), as.Date("2020-09-01"), by="day")

# Search for 300m (333m) LAI within specified date range
( dates.lai <- cgls_urls(dates = d, resolution = 300,
```

```
                          product = "lai") )

# Return all 300m LAI
all.lai <- cgls_urls(resolution = 300, product = "lai")
  nrow(all.lai)


## Not run:
# Example for downloading URL's
# You need to define your login credentials to download data
#   username = "xxxx"
#   password = "xxxx"

  for(i in 1:length(dates.lai)){
    if(i > 1){ Sys.sleep(3) }
    file.url <- paste0("https://", paste(username, password, sep=":"), "@",
                       sub(".*//", "", dates.lai[i]))
      download.file(file.url, file.path(tempdir(),
                    basename(dates.lai[i])), mode = 'wb')
  }

## End(Not run)
```

`chae` *Canine-Human Age Equivalent*

Description

Calculates canines equivalent human age (for fun)

Usage

```
chae(x)
```

Arguments

`x`	numeric vector, dog age

Value

numeric vector, equivalent human age

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

References

Wang, T., J. M, A.N. Hogan, S. Fong, K. Licon et al. (2020) quantitative translation of dog-to-human aging by conserved remodeling of epigenetic networks. Cell Systems 11(2)176-185

Examples

```
dat <- data.frame(DogAge = seq(0,18,0.25),
             HumanAge=chae(seq(0,18,0.25)))[-1,]

plot(dat$DogAge, dat$HumanAge, "l",
     main="Canine-Human Age Equivalence",
 ylab="Human Age", xlab="Dog Age")
  points( 15, chae(15), col="red", pch=19, cex=1.5)
  points( 10, chae(10), col="blue", pch=19, cex=1.5)
  points( 3, chae(3), col="black", pch=19, cex=1.5)
legend("bottomright", legend=c("Camas (15-YO)", "Kele (10-YO)", "Aster (3-YO)"),
      pch=c(19,19,19), cex=c(1.5,1.5,1.5),
     col=c("red","blue","black"))
```

---

chen *Cross-correlation data from Chen (2015)*

Description

Urbanization and economic development data from Chen (2015) compiled from, National Bureau of Statistics of China

Format

A list object with 3 elements:

X per capita GRP(yuan)

Y Level of urbanization percent

M Railway Distance (km) matrix of 29 Chinese regions

Source

https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0126158

References

Chen, Y.G. (2012) On the four types of weight functions for spatial contiguity matrix. Letters in Spatial and Resource Sciences 5(2):65-72

Chen, Y.G. (2013) New approaches for calculating Moran’s index of spatial autocorrelation. PLoS ONE 8(7):e68336

Chen, Y.G. (2015) A New Methodology of Spatial Cross-Correlation Analysis. PLoS One 10(5):e0126158. doi:10.1371/journal.pone.0126158

`classBreaks` *Class breaks*

Description

Finds class breaks in a distribution

Usage

```
classBreaks(x, n, type = c("equal", "quantile", "std", "geometric"))
```

Arguments

`x`	A vector to find breaks for
`n`	Number of breaks
`type`	Statistic used to find breaks c("equal", "quantile", "std", "geometric")

Details

The robust std method uses sqrt(sum(x^2)/(n-1)) to center the data before deriving "pretty" breaks.

Value

A vector containing class break values the length is n+1 to allow for specification of ranges

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
 y <- rnbinom(100, 10, 0.5)
   classBreaks(y, 10)
   classBreaks(y, 10, type="quantile")

opar <- par(no.readonly=TRUE)
   par(mfrow=c(2,2))
     d <- density(y)
     plot(d, type="n", main="Equal Area breaks")
       polygon(d, col="cyan")
       abline(v=classBreaks(y, 10))
     plot(d, type="n", main="Quantile breaks")
       polygon(d, col="cyan")
       abline(v=classBreaks(y, 10, type="quantile"))
     plot(d, type="n", main="Robust Standard Deviation breaks")
       polygon(d, col="cyan")
       abline(v=classBreaks(y, 10, type="std"))
     plot(d, type="n", main="Geometric interval breaks")
       polygon(d, col="cyan")
```

```
      abline(v=classBreaks(y, 10, type="geometric"))
par(opar)

( y.breaks <- classBreaks(y, 10) )
cut(y, y.breaks, include.lowest = TRUE, labels = 1:10)
```

`collinear` *Collinearity test*

Description

Test for linear or nonlinear collinearity/correlation in data

Usage

```
collinear(x, p = 0.85, nonlinear = FALSE, p.value = 0.001)
```

Arguments

`x`	A data.frame or matrix containing continuous data
`p`	The correlation cutoff (default is 0.85)
`nonlinear`	A boolean flag for calculating nonlinear correlations (FALSE/TRUE)
`p.value`	If nonlinear is TRUE, the p value to accept as the significance of the correlation

Details

Evaluation of the pairwise linear correlated variables to remove is accomplished through calculating the mean correlations of each variable and selecting the variable with higher mean. If nonlinear = TRUE, pairwise nonlinear correlations are evaluated by fitting y as a semi-parametrically estimated function of x using a generalized additive model and testing whether or not that functional estimate is constant, which would indicate no relationship between y and x thus, avoiding potentially arbitrary decisions regarding the order in a polynomial regression.

Value

Messages and a vector of correlated variables

Author(s)

Jeffrey S. Evans <jeffrey_evans<at>tnc.org>

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
data(cor.data)

# Evaluate linear correlations on linear dataCollinearity between
head( dat <- cor.data[[4]] )
pairs(dat, pch=20)
  ( cor.vars <- collinear( dat ) )

# Remove identified variable(s)
head( dat[,-which(names(dat) %in% cor.vars)] )


# Evaluate linear correlations on nonlinear data
#   using nonlinear correlation function
plot(cor.data[[1]], pch=20)
  collinear(cor.data[[1]], p=0.80, nonlinear = TRUE )
```

`combine` *raster combine*

Description

Combines rasters into all unique combinations of inputs

Usage

```
combine(x)
```

Arguments

`x`	raster stack/brick or SpatialPixelsDataFrame object

Details

A single ratified raster object is returned with the summary table as the raster attribute table, this is most similar to the ESRI format resulting from their combine function.

Please note that this is not a memory safe function that utilizes out of memory in the manner that the terra package does.

Value

A ratified (factor) terra SpatRaster representing unique combinations.

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
library(terra)

# Create example data (with a few NA's introduced)
 r1 <- rast(nrows=100, ncol=100)
   names(r1) <- "LC1"
   r1[] <- round(runif(ncell(r1), 1,4),0)
     r1[c(8,10,50,100)] <- NA
 r2 <- rast(nrows=100, ncol=100)
   names(r2) <- "LC2"
   r2[] <- round(runif(ncell(r2), 2,6),0)
     r2[c(10,50,100)] <- NA
 r3 <- rast(nrows=100, ncol=100)
   names(r3) <- "LC3"
   r3[] <- round(runif(ncell(r3), 2,6),0)
     r3[c(10,50,100)] <- NA
 r <- c(r1,r2,r3)
   names(r) <- c("LC1","LC2","LC3")

 # Combine rasters with a multilayer stack
 cr <- combine(r)
   head(cr$summary)
   plot(cr$combine)

# or, from separate layers
 cr <- combine(c(r1,r3))
```

`concordance` *Concordance test for binomial models*

Description

Performs a concordance/disconcordance (C-statistic) test on binomial models.

Usage

```
concordance(y, p)
```

Arguments

`y`	vector of binomial response variable used in model
`p`	estimated probabilities from fit binomial model

Details

Test of binomial regression for the hypothesis that probabilities of all positives [1], are greater than the probabilities of the nulls [0]. The concordance would be 100 inverse of concordance, representing the null. The C-statistic has been show to be comparable to the area under an ROC

Results are: concordance - percent of positives that are greater than probabilities of nulls. discordance - concordance inverse of concordance representing the null class, tied - number of tied probabilities and pairs - number of pairs compared

Value

list object with: concordance, discordance, tied and pairs

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

References

Austin, P.C. & E.W. Steyerberg (2012) Interpreting the concordance statistic of a logistic regression model: relation to the variance and odds ratio of a continuous explanatory variable. BMC Medical Research Methodology, 12:82

Harrell, F.E. (2001) Regression modelling strategies. Springer, New York, NY.

Royston, P. & D.G. Altman (2010) Visualizing and assessing discrimination in the logistic regression model. Statistics in Medicine 29(24):2508-2520

Examples

```
data(mtcars)
dat <- subset(mtcars, select=c(mpg, am, vs))
glm.reg <- glm(vs ~ mpg, data = dat, family = binomial)
concordance(dat$vs, predict(glm.reg, type = "response"))
```

`conf.interval` *Confidence interval for mean or median*

Description

Calculates confidence interval for the mean or median of a distribution with unknown population variance

Usage

```
conf.interval(x, cl = 0.95, stat = "mean", std.error = TRUE)
```

Arguments

`x`	Vector to calculate confidence interval for
`cl`	Percent confidence level (default = 0.95)
`stat`	Statistic (mean or median)
`std.error`	Return standard error (TRUE/FALSE)

Value

data.frame contaning:

- lci - Lower confidence interval value
- uci - Upper confidence interval value
- mean - If stat = "mean", mean value of distribution
- mean - Value of the mean or median
- conf.level - Confidence level used for confidence interval
- std.error - If std.error = TRUE standard error of distribution

Author(s)

Jeffrey S. Evans jeffrey_evans@tnc.org

Examples

```
x <- runif(100)
cr <- conf.interval(x, cl = 0.97)
print(cr)

d <- density(x)
plot(d, type="n", main = "PDF with mean and 0.97 confidence interval")
  polygon(d, col="cyan3")
  abline(v=mean(x, na.rm = TRUE), lty = 2)
  segments( x0=cr[["lci"]], y0=mean(d$y), x1=cr[["uci"]],
            y1=mean(d$y), lwd = 2.5,
            col = "black")
 legend("topright", legend = c("mean", "CI"),
        lty = c(2,1), lwd = c(1,2.5))
```

`cor.data` *Various correlation structures*

Description

linear and nonlinear correlated data examples

A list object with various linear and nonlinear correlation structures

Format

A list object with 4 elements containing data.frames:

example 1	two columns with nonlinear wave function relationship
example 2	two columns with simple nonlinear relationship
example 3	two columns with nonlinear multi-level wave function relationship
example 4	4 columns with first two having linear relationship

---

`correlogram` *Correlogram*

Description

Calculates and plots a correlogram

Usage

```
correlogram(x, v, dist = 5000, ns = 99, ...)
```

Arguments

`x`	A sf POINT object
`v`	Test variable in x
`dist`	Distance of correlation lags, if latlong=TRUE units are great circle in kilometers
`ns`	Number of simulations to derive simulation envelope
`...`	Arguments passed to cor ('pearson', 'kendall' or 'spearman')

Value

Plot of correlogram and a list object containing:

- autocorrelation is a data.frame object with the following components
- autocorrelation - Autocorrelation value for each distance lag
- dist - Value of distance lag
- lci - Lower confidence interval (p=0.025)
- uci - Upper confidence interval (p=0.975)
- CorrPlot recordedplot object to recall plot

Author(s)

Jeffrey S. Evans jeffrey_evans@tnc.org

Examples

```
library(sf)
if(require(sp, quietly = TRUE)) {
  data(meuse, package = "sp")
  meuse <- st_as_sf(meuse, coords = c("x", "y"), crs = 28992,
                    agr = "constant")
}

zinc.cg <- correlogram(x = meuse, v = meuse$zinc, dist = 250, ns = 9)
```

`cross.tab` *Class comparison between two nominal rasters*

Description

Creates a labeled cross tabulation between two nominal rasters

Usage

```
cross.tab(x, y, values = NULL, labs = NULL, pct = FALSE, ...)
```

Arguments

`x`	A terra SpatRaster class object
`y`	A terra SpatRaster class object to compare to x
`values`	Expected values in both rasters
`labs`	Labels associated with values argument
`pct`	(TRUE/FALSE) return proportions rather than counts
`...`	Additional arguments

Details

This function returns a cross tabulation between two nominal rasters. Arguments allow for labeling the results and returning proportions rather than counts. It also accounts for asymmetrical classes between the two rasters

Value

a table with the cross tabulated counts

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

References

Pontius Jr, R.G., Shusas, E., McEachern, M. (2004). Detecting important categorical land changes

Examples

```
library(terra)

e <- ext(179407.8, 181087.9, 331134.4, 332332.1)
lulc2010 <- rast(e, resolution=20)
  lulc2010[] <- sample(1:5, ncell(lulc2010), replace=TRUE)
lulc2020 <- rast(e, resolution=20)
  lulc2020[] <- sample(1:5, ncell(lulc2020), replace=TRUE)
```

```
 ( v = sort(unique(c(lulc2010[], lulc2020[]))) )
 l = c("water","urban","forest",
      "ag","barren")

cross.tab(lulc2010, lulc2020)
cross.tab(lulc2010, lulc2020, values = v, labs = l)
cross.tab(lulc2010, lulc2020, values = v, labs = l, pct=TRUE)

# Create asymmetrical classes
na.idx <- which(!is.na(lulc2010[]))
lulc2020[na.idx] <- sample(c(1,2,4,5), length(na.idx), replace=TRUE)
cross.tab(lulc2010, lulc2020, values = v, labs = l, pct=TRUE)
```

`crossCorrelation` *Spatial cross correlation*

Description

Calculates univariate or bivariate spatial cross-correlation using local Moran's-I (LISA), following Chen (2015)

Usage

```
crossCorrelation(
  x,
  y = NULL,
  coords = NULL,
  w = NULL,
  type = c("LSCI", "GSCI"),
  k = 999,
  dist.function = c("inv.power", "neg.exponent", "none"),
  scale.xy = TRUE,
  scale.partial = FALSE,
  scale.matrix = FALSE,
  alpha = 0.05,
  clust = TRUE,
  return.sims = FALSE
)
```

Arguments

`x`	Vector of x response variables
`y`	Vector of y response variables, if not specified the univariate statistic is returned
`coords`	A matrix of coordinates corresponding to (x,y), only used if w = NULL. Can also be an sp object with relevant x,y coordinate slot (ie., points or polygons)

`w`	Spatial neighbors/weights in matrix format. Dimensions must match (n(x),n(y)) and be symmetrical. If w is not defined then a default method is used.
`type`	c("LSCI","GSCI") Return Local Spatial Cross-correlation Index (LSCI) or Global Spatial cross-correlation Index (GSCI)
`k`	Number of simulations for calculating permutation distribution under the null hypothesis of no spatial autocorrelation
`dist.function`	("inv.power", "neg.exponent", "none") If w = NULL, the default method for deriving spatial weights matrix, options are: inverse power or negative exponent, none is for use with a provided matrix
`scale.xy`	(TRUE/FALSE) scale the x,y vectors, if FALSE it is assumed that they are already scaled following Chen (2015)
`scale.partial`	(FALSE/TRUE) rescale partial spatial autocorrelation statistics
`scale.matrix`	(FALSE/TRUE) If a neighbor/distance matrix is passed, should it be scaled using (w/sum(w))
`alpha`	= 0.05 confidence interval (default is 95 pct)
`clust`	(FALSE/TRUE) Return approximated lisa clusters
`return.sims`	(FALSE/TRUE) Return randomizations vector n = k

Details

In specifying a distance matrix, you can pass a coordinates matrix or spatial object to coords or alternately, pass a distance or spatial weights matrix to the w argument. If the w matrix represents spatial weights dist.function="none" should be specified. Otherwise, w is assumed to represent distance and will be converted to spatial weights using inv.power or neg.exponent. The w distances can represent an alternate distance hypothesis (eg., road, stream, network distance) Here are example argument usages for defining a matrix.

- IF coords=x, w=NULL, dist.function= c("inv.power", "neg.exponent") A distance matrix is derived using the data passed to coords then spatial weights derived using one of the dist.function options
- IF cords=NULL, w=x, dist.function= c("inv.power", "neg.exponent") It is expected that the distance matrix specified with w represent some form of distance then the spatial weights are derived using one of the dist.function options
- IF cords=NULL, w=x, dist.function="none" It is assumed that the matrix passed to w already represents the spatial weights

Value

When not simulated k=0, a list containing:

- I - Global autocorrelation statistic
- SCI - - A data.frame with two columns representing the xy and yx autocorrelation
- nsim - value of NULL to represent p values were derived from observed data (k=0)
- p - Probability based observations above/below confidence interval
- t.test - Probability based on t-test

- clusters - If "clust" argument TRUE, vector representing LISA clusters

When simulated (k>0), a list containing:

- I - Global autocorrelation statistic
- SCI - A data.frame with two columns representing the xy and yx autocorrelation
- nsim - value representing number of simulations
- global.p - p-value of global autocorrelation statistic
- local.p - Probability based simulated data using successful rejection of t-test
- range.p - Probability based on range of probabilities resulting from paired t-test
- clusters - If "clust" argument TRUE, vector representing lisa clusters

References

Chen, Y.G. (2012) On the four types of weight functions for spatial contiguity matrix. Letters in Spatial and Resource Sciences 5(2):65-72

Chen, Y.G. (2013) New approaches for calculating Moran's index of spatial autocorrelation. PLoS ONE 8(7):e68336

Chen, Y.G. (2015) A New Methodology of Spatial Cross-Correlation Analysis. PLoS One 10(5):e0126158. doi:10.1371/journal.pone.0126158

Examples

```
# replicate Chen (2015)
 data(chen)
( r <- crossCorrelation(x=chen[["X"]], y=chen[["Y"]], w = chen[["M"]],
                        clust=TRUE, type = "LSCI", k=0,
                        dist.function = "inv.power") )


library(sf)
library(spdep)

  if (require(sp, quietly = TRUE)) {
   data(meuse, package = "sp")
   meuse <- st_as_sf(meuse, coords = c("x", "y"), crs = 28992, agr = "constant")
  }

#### Using a default spatial weights matrix method (inverse power function)
( I <- crossCorrelation(meuse$zinc, meuse$copper,
             coords = st_coordinates(meuse)[,1:2], k=99) )
  meuse$lisa <- I$SCI[,"lsci.xy"]
    plot(meuse["lisa"], pch=20)

#### Providing a distance matrix
if (require(units, quietly = TRUE)) {
  Wij <- units::drop_units(st_distance(meuse))
 ( I <- crossCorrelation(meuse$zinc, meuse$copper, w = Wij, k=99) )
```

```
#### Providing an inverse power function weights matrix
  Wij <- 1 / Wij
    diag(Wij) <- 0
      Wij <- Wij / sum(Wij)
        diag(Wij) <- 0
 ( I <- crossCorrelation(meuse$zinc, meuse$copper, w = Wij,
                         dist.function = "none", k=99) )
}
```

`csi` *Cosine Similarity Index*

Description

Calculates the cosine similarity and angular similarity on two vectors or a matrix

Usage

```
csi(x, y = NULL)
```

Arguments

`x`	A vector or matrix object
`y`	If x is a vector, then a vector object

Details

The cosine similarity index is a measure of similarity between two vectors of an inner product space. This index is bested suited for high-dimensional positive variable space. One useful application of the index is to measure separability of clusters derived from algorithmic approaches (e.g., k-means). It is a good common practice to center the data before calculating the index. It should be noted that the cosine similarity index is mathematically, and often numerically, equivalent to the Pearson's correlation coefficient

The cosine similarity index is derived: s(xy) = x * y / ||x|| * ||y||, where the expected is 1.0 (perfect similarity) to -1.0 (perfect dissimilarity). A normalized angle between the vectors can be used as a bounded similarity function within [0,1] angular similarity = 1 - (cos(s)^-1/pi)

Value

If x is a matrix, a list object with: similarity and angular.similarity matrices or, if x and y are vectors, a vector of similarity and angular.similarity

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
# Compare two vectors (centered using scale)
x=runif(100)
y=runif(100)^2
csi(as.vector(scale(x)),as.vector(scale(y)))

# Compare columns (vectors) in a matrix (centered using scale)
x <- matrix(round(runif(100),0),nrow=20,ncol=5)
( s <- csi(scale(x)) )

# Compare vector (x) to each column in a matrix (y)
y <- matrix(round(runif(500),3),nrow=100,ncol=5)
x=runif(100)
csi(as.vector(scale(x)),scale(y))
```

`curvature` *Surface curvature*

Description

Calculates Zevenbergen & Thorne, McNab's or Bolstad's curvature

Usage

```
curvature(x, type = c("planform", "profile", "total", "mcnab", "bolstad"), ...)
```

Arguments

`x`	A terra SpatRaster object
`type`	Method used c("planform", "profile", "total", "mcnab", "bolstad")
`...`	Additional arguments passed to focal

Details

The planform and profile curvatures are the second derivative(s) of the elevation surface, or the slope of the slope. Profile curvature is in the direction of the maximum slope, and the planform curvature is perpendicular to the direction of the maximum slope. Negative values in the profile curvature indicate the surface is upwardly convex whereas, positive values indicate that the surface is upwardly concave. Positive values in the planform curvature indicate an that the surface is laterally convex whereas, negative values indicate that the surface is laterally concave.

Total curvature is the sigma of the profile and planform curvatures. A value of 0 in profile, planform or total curvature, indicates the surface is flat. The planform, profile and total curvatures are derived using Zevenbergen & Thorne (1987) via a quadratic equation fit to eight neighbors as such, the s (focal window size) argument is ignored.

McNab's and Bolstad's variants of the surface curvature (concavity/convexity) index (McNab 1993; Bolstad & Lillesand 1992; McNab 1989). The index is based on features that confine the view from the center of a 3x3 window. In the Bolstad equation, edge correction is addressed by dividing by the radius distance to the outermost cell (36.2m).

Value

raster class object of surface curvature

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

References

Bolstad, P.V., and T.M. Lillesand (1992). Improved classification of forest vegetation in northern Wisconsin through a rule-based combination of soils, terrain, and Landsat TM data. Forest Science. 38(1):5-20.

Florinsky, I.V. (1998). Accuracy of Local Topographic Variables Derived from Digital Elevation Models. International Journal of Geographical Information Science, 12(1):47-62.

McNab, H.W. (1989). Terrain shape index: quantifying effect of minor landforms on tree height. Forest Science. 35(1):91-104.

McNab, H.W. (1993). A topographic index to quantify the effect of mesoscale landform on site productivity. Canadian Journal of Forest Research. 23:1100-1107.

Zevenbergen, L.W. & C.R. Thorne (1987). Quantitative Analysis of Land Surface Topography. Earth Surface Processes and Landforms, 12:47-56.

Examples

```
library(terra)
elev <- rast(system.file("extdata/elev.tif", package="spatialEco"))

crv <- curvature(elev, type = "planform")
mcnab.crv <- curvature(elev, type = "mcnab")
    plot(mcnab.crv, main="McNab's curvature")
```

`dahi` *Diurnal Anisotropic Heat Index*

Description

Simple approximation of the anisotropic diurnal heat (Ha) distribution

Usage

```
dahi(x, amax = 202.5)
```

Arguments

`x`	An elevation raster of class terra SpatRaster
`amax`	The Alpha Max (amax) parameter in degrees defined as: minimum = 0, maximum = 360 with the default = 202.500

Details

The Diurnal Anisotropic Heat Index is based on this equation. Ha = cos(amax - a) * arctan(b) Where; amax defines the aspect with the maximum total heat surplus, a is the aspect and b is the slope angle.

Value

terra SpatRaster class object Diurnal Anisotropic Heat Index

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

References

Boehner, J., and Antonic, O. (2009) Land-surface parameters specific to topo-climatology. In: Hengl, T., & Reuter, H. (Eds.), Geomorphometry - Concepts, Software, Applications. Developments in Soil Science, 33:195-226

Examples

```
library(terra)
elev <- rast(system.file("extdata/elev.tif", package="spatialEco"))
Ha <- dahi(elev)
  plot(Ha)
```

`date_seq` *date sequence*

Description

creates date sequence given start and stop dates

Usage

```
date_seq(
  start,
  end,
  step = c("day", "week", "month", "quarter", "year", "minute"),
  rm.leap = FALSE
)
```

Arguments

`start`	Start date in "yyyy/mm/dd" character format
`end`	End date in "yyyy/mm/dd" character format
`step`	Time step, options are c("day", "week", "month", "quarter", "year", "minute")
`rm.leap`	Remove extra days in leap years

Value

A date vector of class POSIXct for minute and Date for other options

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
# monthly steps 1990/01/01 - 2019/12/31
d <- date_seq("1990/01/01", "2019/12/31", step="month")

# daily steps 1990/01/01 - 2019/12/31
d <- date_seq("1990/01/01", "2019/12/31", step="day")

# daily steps 1990/01/01 - 2019/12/31 with leap days removed
d <- date_seq("1990/01/01", "2019/12/31", step="day", rm.leap=TRUE)

# daily step 2008/12/29 - 2008/12/31, 2008 is leap year
d <- date_seq("2008/12/29", "2008/12/31")

# minutes step 2008/12/29 - 2008/12/31, 2008 is leap year
d <- date_seq("2008/12/29", "2008/12/31", step="minute")
```

`daymet.point` *DAYMET point values*

Description

Downloads DAYMET climate variables for specified point and time-period

Usage

```
daymet.point(
  lat,
  long,
  start.year,
  end.year,
  site = NULL,
```

```
  files = FALSE,
  echo = FALSE
)
```

Arguments

`lat`	latitude of point (decimal degrees WGS84)
`long`	longitude pf point (decimal degrees WGS84)
`start.year`	First year of data
`end.year`	Last year of data
`site`	Unique identification value that is appended to data
`files`	(TRUE/FALSE) Write file to disk
`echo`	(TRUE/FALSE) Echo progress

Details

data is available for Long -131.0 W and -53.0 W; lat 52.0 N and 14.5 N Function uses the Single Pixel Extraction tool and returns year, yday, dayl(s), prcp (mm/day), srad (W/m^2), swe (kg/m^2), tmax (deg c), tmin (deg c), vp (Pa) Metadata for DAYMET single pixel extraction: `https://daymet.ornl.gov/files/UserGuides/current/readme_singlepointextraction.pdf`

Value

A data.frame with geographic coordinate point-level climate results

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
( d <- daymet.point(lat = 36.0133, long = -84.2625, start.year = 2013,
                    end.year=2014, site = "1", files = FALSE, echo = FALSE) )
```

`daymet.tiles` *DAYMET Tile ID's*

Description

Returns a vector of DAYMET tile id's within a specified extent

Usage

```
daymet.tiles(...)
```

Arguments

`...`	ignored

Value

Vector of DAYMET tile IDS or if sp = TRUE a sp class SpatialPolygonsDataFrame

Note

Function accepts sp, raster or extent class object or bounding coordinates. All input must be in the same projection as the tile index SpatialPolygonsDataFrame. The library includes the DAYMAT tile index "DAYMET_tiles" which can be add using data(), see examples.

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

`dispersion` *Dispersion (H-prime)*

Description

Calculates the dispersion ("rarity") of targets associated with planning units

Usage

```
dispersion(x)
```

Arguments

`x`	data.frame object of target values

Details

The dispersion index (H-prime) is calculated H = sum(sqrt(p) / sqrt(a)) where; P = (sum of target in planning unit / sum of target across all planning units) and a = (count of planning units containing target / number of planning units)

Value

data.frame with columns H values for each target, H , sH, sHmax

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

References

Evans, J.S., S.R. Schill, G.T. Raber (2015) A Systematic Framework for Spatial Conservation Planning and Ecological Priority Design in St. Lucia, Eastern Caribbean. Chapter 26 in Central American Biodiversity : Conservation, Ecology and a Sustainable Future. F. Huettman (eds). Springer, NY.

Examples

```
library(sf)
  data(pu)

d <- dispersion(st_drop_geometry(pu[,2:ncol(pu)]))
  p <- d[,"H"]
clr <- c("#3288BD", "#99D594", "#E6F598", "#FEE08B",
         "#FC8D59", "#D53E4F")
clrs <- ifelse(p < 0.5524462, clr[1],
          ifelse(p >= 0.5524462 & p < 1.223523, clr[2],
            ifelse(p >= 1.223523 & p < 2.465613, clr[3],
           ifelse(p >= 2.465613 & p < 4.76429, clr[4],
             ifelse(p >= 4.76429 & p < 8.817699, clr[5],
               ifelse(p >= 8.817699, clr[6], NA))))))
plot(st_geometry(pu), col=clrs, border=NA)
  legend("bottomleft", legend=rev(c("Very Rare","Rare","Moderately Rare",
         "Somewhat Common","Common","Over Dispersed")),
         fill=clr, cex=0.6, bty="n")
  box()
```

`dissection` *Dissection*

Description

Calculates the Evans (1972) Martonne’s modified dissection

Usage

```
dissection(x, s = 5, ...)
```

Arguments

`x`	A terra SpatRaster class object
`s`	Focal window size
`...`	Additional arguments passed to terra::lapp

Details

Dissection is calculated as: (z(s) - min(z(s))) / (max(z(s)) - min(z(s)))

Value

A SpatRaster class object of Martonne's modified dissection

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
library(terra)
elev <- rast(system.file("extdata/elev.tif", package="spatialEco"))
  d <- dissection(elev, s=3)
    plot(d, main="dissection")
```

`divergence` *divergence*

Description

Kullback-Leibler Divergence (Cross-entropy)

Usage

```
divergence(x, y, type = c("Kullback-Leibler", "cross-entropy"))
```

Arguments

`x`	a vector of integer values, defining observed
`y`	a vector of integer values, defining estimates
`type`	Type of divergence statistic c("Kullback-Leibler", "cross-entropy")

Value

single value vector with divergence statistic

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
x <- round(runif(10,1,4),0)
y <- round(runif(10,1,4),0)

divergence(x, y)
divergence(x, y, type = "cross-entropy")
```

`effect.size` *Cohen's-d effect size*

Description

Cohen's-d effect size with pooled sd for a control and experimental group

Usage

```
effect.size(y, x, pooled = TRUE, conf.level = 0.95)
```

Arguments

`y`	A character or factor vector
`x`	A numeric vector, same length as y
`pooled`	Pooled or population standard deviation (TRUE/FALSE)
`conf.level`	Specified confidence interval. Default is 0.95

Value

An effect.size class object with x, y and a data.frame with columns for effect size, lower confidence interval, lower confidence interval. The row names of the data frame represent the levels in y

Note

This implementation will iterate through each class in y and treating a given class as the experimental group and all other classes as a control case. Each class had d and the confidence interval derived. A negative d indicate directionality with same magnitude. The expected range for d is 0 - 3 d is derived; (mean(experimental group) - mean(control group)) / sigma(p) pooled standard deviation is derived; sqrt(((Ne - 1) * sigma(e)^2 + (Nc - 1) * sigma(c)^2) / (Ne + Nc - 2)) where; Ne, Nc = n of experimental and control groups.

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

References

Cohen, J., (1988) Statistical Power Analysis for the Behavioral Sciences (second ed.). Lawrence Erlbaum Associates.

Cohen, J (1992) A power primer. Psychological Bulletin 112(1):155-159

Examples

```
( es <- effect.size(iris$Species, iris$Sepal.Length) )
  plot(es)
```

`elev` *Elevation raster*

Description

elevation raster of Switzerland

Format

A raster RasterLayer class object:

resoultion 5 arc-minute 0.00833 (10000m)

nrow 264

ncol 564

ncell 148896

xmin 5.9

xmax 10.6

ymin 45.7

ymax 47.9

proj4string +proj=longlat +ellps=WGS84

Source

`http://www.diva-gis.org/Data`

`erase.point` *Erase points*

Description

Removes points intersecting a polygon feature class

Usage

```
erase.point(y, x, inside = TRUE)
```

Arguments

`y`	A sf POINT object
`x`	A sf POLYGON object
`inside`	(TRUE/FALSE) Remove points inside polygon, else outside polygon

Details

Used to erase points that intersect polygon(s). The default of inside=TRUE erases points inside the polygons however, if inside=FALSE then the function results in an intersection where points that intersect the polygon are retained.

Value

An sf POINT object

Author(s)

Jeffrey S. Evans <jeffrey_evans<at>tnc.org>

Examples

```
library(sf)

if (require(sp, quietly = TRUE)) {
  data(meuse, package = "sp")
  meuse <- st_as_sf(meuse, coords = c("x", "y"), crs = 28992, agr = "constant")

  s <- st_as_sf(st_sample(st_as_sfc(st_bbox(meuse)), size=1000,
               type = "regular"))
    s$id <- 1:nrow(s)
  b <- st_buffer(s[sample(1:nrow(s),5),], dist=300)
    b$id <- 1:nrow(b)

# Erase points based on polygons
in.erase <- erase.point(s, b)
```

```
out.erase <- erase.point(s, b, inside = FALSE)

 opar <- par(no.readonly=TRUE)
 par(mfrow=c(2,2))
   plot(st_geometry(s), pch=20, main="original data")
     plot(st_geometry(b),add=TRUE)
   plot(st_geometry(in.erase), pch=20, main="erased data")
     plot(st_geometry(b),add=TRUE)
   plot(st_geometry(out.erase), pch=20,
        main="erased data using inside=FALSE")
     plot(st_geometry(b),add=TRUE)
 par(opar)

} else {
  cat("Please install sp package to run example", "\n")
}
```

`extract.vertices` *Extract vertices for polygons or lines*

Description

Extracts [x,y] vertices from an sf line or polygon object

Usage

```
extract.vertices(x, join = TRUE)
```

Arguments

`x`	An sf line or polygon class object
`join`	(TRUE/FALSE) Joint attributes from original object

Details

This function returns the vertices of a line or polygon object, as opposed to the polygon centroids or line start/stop coordinates

Value

An sf POINT object of extrated line or polygon vertices

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
library(sf)
nc <- sf::st_read(system.file("shape/nc.shp", package="sf"))
  nc <- suppressWarnings(sf::st_cast(nc, "POLYGON"))
    nc <- nc[c(10,50),]

( v <- extract.vertices(nc) )
  plot(st_geometry(nc))
    plot(st_geometry(v), pch=20, cex=2, col="red", add=TRUE)
```

`fuzzySum` *Fuzzy Sum*

Description

Calculates the fuzzy sum of a vector

Usage

```
fuzzySum(x)
```

Arguments

`x`	Vector of values to apply fuzzy sum

Details

The fuzzy sum is an increasing linear combination of values. This can be used to sum probabilities or results of multiple density functions.

Value

Value of fuzzy sum

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
p = c(0.8,0.76,0.87)
  fuzzySum(p)
  sum(p)

p = c(0.3,0.2,0.1)
  fuzzySum(p)
  sum(p)
```

`gaussian.kernel` *Gaussian Kernel*

Description

Creates a Gaussian Kernel of specified size and sigma

Usage

```
gaussian.kernel(sigma = 2, s = 5)
```

Arguments

`sigma`	sigma (standard deviation) of kernel (defaults 2)
`s`	scale defining the number of rows and columns for kernel (default 5)

Value

Symmetrical (NxN) matrix of a Gaussian distribution

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
opar <- par()
  par(mfrow=c(2,2))
  persp(gaussian.kernel(sigma=1, s=27), theta = 135,
        phi = 30, col = "grey", ltheta = -120, shade = 0.6,
        border=NA )
  persp(gaussian.kernel(sigma=2, s=27), theta = 135, phi = 30,
        col = "grey", ltheta = -120, shade = 0.6, border=NA )
  persp(gaussian.kernel(sigma=3, s=27), theta = 135, phi = 30,
        col = "grey", ltheta = -120, shade = 0.6, border=NA )
  persp(gaussian.kernel(sigma=4, s=27), theta = 135, phi = 30,
        col = "grey", ltheta = -120, shade = 0.6, border=NA )
 par(opar)
```

`geo.buffer` *Buffer geographic data*

Description

Buffers data in geographic (Latitude/Longitude) projection

Usage

```
geo.buffer(x, r, ...)
```

Arguments

`x`	A sf or sp vector class object
`r`	Buffer radius in meters
`...`	Additional arguments passed to sf::st_buffer

Details

Projects (Latitude/Longitude) data in decimal-degree geographic projection using an on-the-fly azimuthal equidistant projection in meters centered on

Value

an sp or sf polygon class object representing buffer for each feature

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

See Also

`st_buffer` for st_buffer ... arguments

Examples

```
library(sf)
e <- c(61.87125, 23.90153, 76.64458, 37.27042)
  names(e) <- c("xmin", "ymin", "xmax", "ymax")
  s <- st_as_sf(st_sample(st_as_sfc(st_bbox(e)), size=100,
                 type = "regular"))
st_crs(s) <- st_crs(4326)
      s$id <- 1:nrow(s)

b <- geo.buffer(x=s, r=1000)
  plot(st_geometry(b[1,]))
     plot(st_geometry(s[1,]), pch=20,cex=2, add=TRUE)
```

`group.pdf` *Probability density plot by group*

Description

Creates a probability density plot of y for each group of x

Usage

```
group.pdf(
  x,
  y,
  col = NULL,
  lty = NULL,
  lwd = NULL,
  lx = "topleft",
  ly = NULL,
  ...
)
```

Arguments

`x`	Numeric, character or factorial vector of grouping variable (must be same length as y)
`y`	Numeric vector (density variable)
`col`	Optional line colors (see par, col)
`lty`	Optional line types (see par, lty)
`lwd`	Optional line widths (see par, lwd)
`lx`	Position of legend (x coordinate or 'topright', 'topleft', 'bottomright', 'bottomleft')
`ly`	Position of legend (y coordinate)
`...`	Additional arguments passed to plot

Value

Plot of grouped PDF's

Author(s)

Jeffrey S. Evans <jeffrey_evans<at>tnc.org>

References

Simonoff, J. S. (1996). Smoothing Methods in Statistics. Springer-Verlag, New York.

Examples

```
y=dnorm(runif(100))
x=rep(c(1,2,3), length.out=length(y))
group.pdf(x=as.factor(x), y=y, main='Probability Density of y by group(x)',
ylab='PDF', xlab='Y', lty=c(1,2,3))
```

`hexagons` *Hexagons*

Description

Create hexagon polygons

Usage

```
hexagons(x, res = 100)
```

Arguments

`x`	sf class object indicating extent
`res`	Area of resulting hexagons

Details

Based on extent of x, creates a hexagon mesh with size of hexagons defined by res argumnet

Value

sf POLYGONS object

Examples

```
library(sf)
if(require(sp, quietly = TRUE)) {
  data(meuse, package = "sp")
  meuse <- st_as_sf(meuse, coords = c("x", "y"), crs = 28992,
                    agr = "constant")

hex <- hexagons(meuse, res=300)
  plot(st_geometry(hex))
    plot(st_geometry(meuse),pch=20,add=TRUE)

# subset hexagons to intersection with points
idx <- which(apply(st_intersects(hex, meuse, sparse=FALSE), 1, any))
hex.sub <- hex[idx,]
  plot(st_geometry(hex.sub))
    plot(st_geometry(meuse),pch=20,add=TRUE)
```

```
} else {
  cat("Please install sp package to run example", "\n")
}
```

`hli` *Heat Load Index*

Description

Calculates the McCune & Keon (2002) Heat Load Index

Usage

```
hli(x, check = TRUE, force.hemisphere = c("none", "southern", "northern"))
```

Arguments

`x`	terra SpatRaster class object
`check`	(TRUE/FALSE) check for projection integrity and calculate central latitude for non-geographic projections
`force.hemisphere`	If country is split at the equator, force southern or northern hemisphere equation c("southern", "northern")

Details

Describes A southwest facing slope should have warmer temperatures than a southeast facing slope, even though the amount of solar radiation they receive is equivalent. The McCune and Keon (2002) method accounts for this by "folding" the aspect so that the highest values are southwest and the lowest values are northeast. Additionally, this method account for steepness of slope, which is not addressed in most other aspect rescaling equations. HLI values range from 0 (coolest) to 1 (hottest).

The equations follow McCune (2007) and support northern and southern hemisphere calculations. The folded aspect for northern hemispheres use (180 - (Aspect – 225)) and for Southern hemisphere (180 - (Aspect – 315)). If a country is split at the equator you can use the force.hemisphere argument to choose which equation to use. Valid values for this argument are "southern" and "northern" with the default "none".

Value

terra SpatRaster class object of McCune & Keon (2002) Heat Load Index

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

References

McCune, B., and D. Keon (2002) Equations for potential annual direct incident radiation and heat load index. Journal of Vegetation Science. 13:603-606.

McCune, B. (2007). Improved estimates of incident radiation and heat load using non-parametric regression against topographic variables. Journal of Vegetation Science 18:751-754.

Examples

```
library(terra)
elev <- rast(system.file("extdata/elev.tif", package="spatialEco"))
heat.load <- hli(elev)
  plot(heat.load, main="Heat Load Index")
```

`hli.pt` *Point estimate of Heat Load Index*

Description

Calculates the McCune & Keon (2002) Heat Load Index

Usage

```
hli.pt(
  alpha,
  theta,
  latitude,
  direct = FALSE,
  scaled = TRUE,
  force.hemisphere = c("none", "southern", "northern")
)
```

Arguments

`alpha`	Aspect in degrees
`theta`	Slope in degrees
`latitude`	A latitude representing the centrality of the data
`direct`	Boolean (FALSE/TRUE) Return direct incident radiation else HLI (default)
`scaled`	Boolean (TRUE/FALSE) Apply arithmetic scale using EXP(h)
`force.hemisphere`	If country is split at the equator, force southern or northern hemisphere equation c("southern", "northern")

Details

Describes A southwest facing slope should have warmer temperatures than a southeast facing slope, even though the amount of solar radiation they receive is equivalent. The McCune and Keon (2002) method accounts for this by "folding" the aspect so that the highest values are southwest and the lowest values are northeast. Additionally, this method account for steepness of slope, which is not addressed in most other aspect rescaling equations. HLI values range from 0 (coolest) to 1 (hottest).

The equations follow McCune (2007) and support northern and southern hemisphere calculations. The folded aspect for northern hemispheres use (180 - (Aspect – 225)) and for Southern hemisphere (180 - (Aspect – 315)). If a country is split at the equator you can use the force.hemisphere argument to choose which equation to use. Valid values for this argument are "southern" and "northern" with the default "none".

Value

Vector of McCune & Keon (2002) Heat Load Index

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

References

McCune, B., and D. Keon (2002) Equations for potential annual direct incident radiation and heat load index. Journal of Vegetation Science. 13:603-606.

McCune, B. (2007). Improved estimates of incident radiation and heat load using non-parametric regression against topographic variables. Journal of Vegetation Science 18:751-754.

Examples

```
# Single point input
hli.pt(theta=180, alpha=30, latitude=40)

# Multiple input, returns results from
#   McCune, B., and D. Keon (2002)
# Raw -0.2551 -0.6280 0.0538 -0.6760 -1.1401 -0.2215
# arithmetic scale 0.7748 0.5337 1.0553 0.5086 0.3198 0.8013

slp = c(0, 30, 30, 0, 30, 30)
asp =c(0, 0, 180, 0, 0, 180)
lat =c(40, 40, 40, 60, 60, 60)
hli.pt(theta = slp, alpha = asp, latitude = lat)
```

hsp *Hierarchical Slope Position*

Description

Calculates a hierarchical scale decomposition of topographic position index

Usage

```
hsp(
  x,
  min.scale = 3,
  max.scale = 27,
  inc = 4,
  win = "rectangle",
  normalize = FALSE
)
```

Arguments

`x`	A terra SpatRaster class object
`min.scale`	Minimum scale (window size)
`max.scale`	Maximum scale (window size)
`inc`	Increment to increase scales
`win`	Window type, options are "rectangle" or "circle"
`normalize`	Normalize results to 0-1 scale (FALSE | TRUE)

Details

if win = "circle" units are distance, if win = "rectangle" units are number of cells

Value

terra SpatRaster class object of slope position

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

References

Murphy M.A., J.S. Evans, and A.S. Storfer (2010) Quantify Bufo boreas connectivity in Yellowstone National Park with landscape genetics. Ecology 91:252-261

Examples

```
library(terra)
elev <- rast(system.file("extdata/elev.tif", package="spatialEco"))
hsp27 <- hsp(elev, 3, 27, 4, normalize = TRUE)
plot(hsp27)
```

`hybrid.kmeans` *Hybrid K-means*

Description

Hybrid K-means clustering using hierarchical clustering to define cluster-centers

Usage

```
hybrid.kmeans(x, k = 2, hmethod = "ward.D", stat = mean, ...)
```

Arguments

`x`	A data.frame or matrix with data to be clustered
`k`	Number of clusters
`hmethod`	The agglomeration method used in hclust
`stat`	The statistic to aggregate class centers (mean or median)
`...`	Additional arguments passed to `kmeans`

Details

This method uses hierarchical clustering to define the cluster-centers in the K-means clustering algorithm. This mitigates some of the know convergence issues in K-means.

Value

returns an object of class "kmeans" which has a print and a fitted method

Note

options for hmethod are: "ward.D", "ward.D2", "single", "complete", "average", mcquitty", "median", "centroid"

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

References

Singh, H., & K. Kaur (2013) New Method for Finding Initial Cluster Centroids in K-means Algorithm. International Journal of Computer Application. 74(6):27-30

Ward, J.H., (1963) Hierarchical grouping to optimize an objective function. Journal of the American Statistical Association. 58:236-24

See Also

`kmeans` for available ... arguments and function details

`hclust` for details on hierarchical clustering

Examples

```
x <- rbind(matrix(rnorm(100, sd = 0.3), ncol = 2),
           matrix(rnorm(100, mean = 1, sd = 0.3), ncol = 2))

# Compare k-means to hybrid k-means with k=4
km <- kmeans(x, 4)
hkm <- hybrid.kmeans(x,k=4)

opar <- par(no.readonly=TRUE)
par(mfrow=c(1,2))
  plot(x[,1],x[,2], col=km$cluster,pch=19, main="K-means")
  plot(x[,1],x[,2], col=hkm$cluster,pch=19, main="Hybrid K-means")
par(opar)
```

`idw.smoothing` *Inverse Distance Weighted smoothing*

Description

Distance weighted smoothing of a variable in a spatial point object

Usage

```
idw.smoothing(x, y, d, k)
```

Arguments

x	An sf POINT class object
y	Numeric data column in x to be smoothed
d	Distance constraint
k	Maximum number of k-nearest neighbors within d

Details

Smoothing is conducted with a weighted-mean where; weights represent inverse standardized distance lags Distance-based or neighbour-based smoothing can be specified by setting the desired neighbour smoothing method to a specified value then the other parameter to the potential maximum. For example; a constraint distance, including all neighbors within 1000 (d=1000) would require k to equal all of the potential neighbors (n-1 or k=nrow(x)-1).

Value

A vector, same length as nrow(x), of smoothed y values

Examples

```
library(sf)
if(require(sp, quietly = TRUE)) {
  data(meuse, package = "sp")
  meuse <- st_as_sf(meuse, coords = c("x", "y"), crs = 28992,
                    agr = "constant")

 # Calculate distance weighted mean on cadmium variable in meuse data
   cadmium.idw <- idw.smoothing(meuse, 'cadmium', k=nrow(meuse), d = 1000)
   meuse$cadmium.wm <- cadmium.idw

   opar <- par(no.readonly=TRUE)
     par(mfrow=c(2,1))
       plot(density(meuse$cadmium), main='Cadmium')
       plot(density(meuse$cadmium.wm), main='IDW Cadmium')
   par(opar)

plot(meuse[c("cadmium","cadmium.wm")], pch=20)

} else {
  cat("Please install sp package to run example", "\n")
}
```

`impute.loess` *Impute loess*

Description

Imputes missing data or smooths using Loess regression

Usage

```
impute.loess(y, s = 0.2, smooth = FALSE)
```

Arguments

`y`	A vector to impute
`s`	Smoothing parameter ()
`smooth`	(FALSE/TRUE) Smooth data, else only replace NA's

Details

Performs a local polynomial regression to smooth data or to impute NA values. The minimal number of non-NA observations to reliably impute/smooth values is 6. There is not a reliably way to impute NA's on the tails of the distributions so if the missing data is in the first or last position of the vector it will remain NA. Please note that smooth needs to be TRUE to return a smoothed vector, else only NA's will be imputed.

Value

A vector the same length as x with NA values filled or the data smoothed (or both).

Author(s)

Jeffrey S. Evans <jeffrey_evans<at>tnc.org>

Examples

```
data(cor.data)
d <- cor.data[[1]][,2]
  plot(d, type="l")
  lines(impute.loess(d, s=0.3, smooth=TRUE), lwd=2, col="red")

# add some NA's
d <- d[1:100]
  d[sample(30:70, 5)] <- NA
  d

impute.loess(d, s=0.2)
```

`insert` *Insert a row or column into a data.frame*

Description

Inserts a new row or column into a data.frame at a specified location

Usage

```
insert(x, MARGIN = 1, value = NULL, idx, name = NULL)
```

Arguments

`x`	Existing data.frame
`MARGIN`	Insert a 1 = row or 2 = column
`value`	A vector of values equal to the length of MARGIN, if nothing specified values with be NA
`idx`	Index position to insert row or column
`name`	Name of new column (not used for rows, MARGIN=1)

Details

Where there are methods to easily add a row/column to the end or beginning of a data.frame, it is not straight forward to insert data at a specific location within the data.frame. This function allows for inserting a vector at a specific location eg., between columns or rows 1 and 2 where row/column 2 is moved to the 3rd position and a new vector of values is inserted into the 2nd position.

Value

A data.frame with the new row or column inserted

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
d <- data.frame(ID=1:10, y=runif(10))

# insert row
insert(d, idx=2)
insert(d, value=c(20,0), idx=2)

# insert column
insert(d, MARGIN=2, idx=2)
insert(d, MARGIN = 2, value = rep(0,10), idx=2, name="x")
```

`insert.values` *Insert Values*

Description

Inserts new values into a vector at specified positions

Usage

```
insert.values(x, value, index)
```

Arguments

`x`	A vector to insert values
`value`	Values to insert into x
`index`	Index position(s) to insert y values into x

Details

This function inserts new values at specified positions in a vector. It does not replace existing values. If a single value is provided for y and l represents multiple positions y will be replicated for the length of l. In this way you can insert the same value at multiple locations.

Value

A vector with values of y inserted into x and the position(s) defined by the index

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
(x=1:10)

 # Insert single value in one location
 insert.values(x, 100, 2)

 # Insert multiple values in multiple locations
 insert.values(x, c(100,200), c(2,8))

 # Insert single value in multiple locations
 insert.values(x, NA, c(2,8))
```

`is.empty` *is.empty*

Description

evaluates empty elements in a vector

Usage

```
is.empty(x, all.na = FALSE, na.empty = TRUE, trim = TRUE)
```

Arguments

`x`	A vector to evaluate elements
`all.na`	(FALSE / TRUE) Return a TRUE if all elements are NA
`na.empty`	(TRUE / FALSE) Return TRUE if element is NA
`trim`	(TRUE / FALSE) Trim empty strings

Details

This function evaluates if an element in a vector is empty the na.empty argument allows for evaluating NA values (TRUE if NA) and all.na returns a TRUE if all elements are NA. The trim argument trims a character string to account for the fact that c(" ") is not empty but, a vector with c("") is empty. Using trim = TRUE will force both to return TRUE

Value

A Boolean indicating empty elements in a vector, if all.na = FALSE a TRUE/FALSE value will be returned for each element in the vector

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
is.empty( c("") )
is.empty( c(" ") )
is.empty( c(" "), trim=FALSE )

is.empty( c("",NA,1) )
is.empty( c("",NA,1), na.empty=FALSE)

is.empty( c(NA,NA,NA) )
is.empty( c(NA,NA,NA), all.na=TRUE )
is.empty( c(NA,2,NA), all.na=TRUE )

any( is.empty( c("",2,3) ) )
any( is.empty( c(1,2,3) ) )
```

`kendall` *Kendall tau trend with continuity correction for time-series*

Description

Calculates a nonparametric statistic for a monotonic trend based on the Kendall tau statistic and the Theil-Sen slope modification

Usage

```
kendall(
  y,
  tau = TRUE,
  intercept = TRUE,
  p.value = TRUE,
  confidence = TRUE,
  method = c("zhang", "yuepilon", "none"),
  threshold = 6,
  ...
)
```

Arguments

`y`	A vector representing a timeseries with >= 8 obs
`tau`	(FALSE/TRUE) return tau values
`intercept`	(FALSE/TRUE) return intercept values
`p.value`	(FALSE/TRUE) return p.values
`confidence`	(FALSE/TRUE) return 95 pct confidence levels
`method`	Method for deriving tau and slope ("zhang", "yuepilon", "none")
`threshold`	The threshold for number of minimum observations in the time-series
`...`	Not used

Details

This function implements Kendall's nonparametric test for a monotonic trend using the Theil-Sen (Theil 1950; Sen 1968; Siegel 1982) method to estimate the slope and related confidence intervals. Critical values are $Z > 1.96$ representing a significant increasing trend and a $Z < -1.96$ a significant decreasing trend ($p < 0.05$). The null hypothesis can be rejected if Tau = 0. Autocorrelation in the time-series is addressed using a prewhitened linear trend following the Zhang et al., (2000) or Yue & Pilon (2002) methods. If you do not have autocorrelation in the data, the "none" or "yuepilon" method is recommended. Please note that changing the threshold to fewer than 6 observations (ideally 8) may prevent the function from failing but, will likely invalidate the statistic. A threshold of <=4 will yield all NA values. If method= "none" a modification of the EnvStats::kendallTrendTest code is implemented.

Value

Depending on arguments, a vector containing:

- Theil-Sen slope, always returned
- Kendall's tau two-sided test, if tau TRUE
- intercept for trend if intercept TRUE
- p value for trend fit if p.value TRUE
- lower confidence level at 95-pct if confidence TRUE
- upper confidence level at 95-pct if confidence TRUE

Author(s)

Jeffrey S. Evans jeffrey_evans@tnc.org

References

Theil, H. (1950) A rank invariant method for linear and polynomial regression analysis. Nederl. Akad. Wetensch. Proc. Ser. A 53:386-392 (Part I), 53:521-525 (Part II), 53:1397-1412 (Part III).

Sen, P.K. (1968) Estimates of Regression Coefficient Based on Kendall's tau. Journal of the American Statistical Association. 63(324):1379-1389.

Siegel, A.F. (1982) Robust Regression Using Repeated Medians. Biometrika, 69(1):242-244

Yue, S., P. Pilon, B. Phinney and G. Cavadias, (2002) The influence of autocorrelation on the ability to detect trend in hydrological series. Hydrological Processes, 16: 1807-1829.

Zhang, X., Vincent, L.A., Hogg, W.D. and Niitsoo, A., (2000) Temperature and Precipitation Trends in Canada during the 20th Century. Atmosphere-Ocean 38(3): 395-429.

See Also

`zyp.trend.vector` for model details

Examples

```
data(EuStockMarkets)
d <- as.vector(EuStockMarkets[,1])
kendall(d)
```

`kl.divergence` *Kullback-Leibler divergence (relative entropy)*

Description

Calculates the Kullback-Leibler divergence (relative entropy)

Usage

```
kl.divergence(object, eps = 10^-4, overlap = TRUE)
```

Arguments

`object`	Matrix or dataframe object with >=2 columns
`eps`	Probabilities below this threshold are replaced by this threshold for numerical stability.
`overlap`	Logical, do not determine the KL divergence for those pairs where for each point at least one of the densities has a value smaller than eps.

Details

Calculates the Kullback-Leibler divergence (relative entropy) between unweighted theoretical component distributions. Divergence is calculated as: int [f(x) (log f(x) - log g(x)) dx] for distributions with densities f() and g().

Value

pairwise Kullback-Leibler divergence index (matrix)

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

References

Kullback S., and R. A. Leibler (1951) On information and sufficiency. The Annals of Mathematical Statistics 22(1):79-86

Examples

```
x <- seq(-3, 3, length=200)
y <- cbind(n=dnorm(x), t=dt(x, df=10))
  matplot(x, y, type='l')
    kl.divergence(y)

# extract value for last column
  kl.divergence(y[,1:2])[3:3]
```

`knn` *Spatial K nearest neighbor*

Description

Find K nearest neighbors for two spatial objects

Usage

```
knn(
  y,
  x,
  k = 1,
  d = NULL,
  ids = NULL,
  weights.y = NULL,
  weights.x = NULL,
  indexes = FALSE
)
```

Arguments

`y`	Spatial sf object or coordinates matrix
`x`	Spatial points or polygons object or coordinates matrix
`k`	Number of neighbors
`d`	Optional search radius
`ids`	Optional column of ID's in x
`weights.y`	A vector or matrix representing covariates of y
`weights.x`	A vector or matrix representing covariates of x
`indexes`	(FALSE/TRUE) Return row indexes of x neighbors

Details

Finds nearest neighbor in x based on y and returns rownames, index and distance, If ids is NULL, rownames of x are returned. If coordinate matrix provided, columns need to be ordered [X,Y]. If a radius for d is specified than a maximum search radius is imposed. If no neighbor is found, a neighbor is not returned

You can specify weights to act as covariates for x and y. The vectors or matrices must match row dimensions with x and y as well as columns matching between weights. In other words, the covariates must match and be numeric.

Value

A data.frame with row indexes (optional), rownames, ids (optional) and distance of k

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

See Also

`nn2` for details on search algorithm

Examples

```
if(require(sp, quietly = TRUE)) {
library(sf)
  data(meuse, package = "sp")
  meuse <- st_as_sf(meuse, coords = c("x", "y"), crs = 28992,
                    agr = "constant")

# create reference and target obs
idx <- sample(1:nrow(meuse), 10)
  pts <- meuse[idx,]
    meuse <- meuse[-idx,]
    meuse$IDS <- 1:nrow(meuse)

# Find 2 neighbors in meuse
```

```
( nn <- knn(pts, meuse, k=2, ids = "IDS", indexes = TRUE) )
  plot( st_geometry(pts), pch=19, main="KNN")
    plot(st_geometry(meuse[nn[,1],]), pch=19, col="red", add=TRUE)

# Using covariates (weights)
wx = as.matrix(st_drop_geometry(meuse[,1:3]))
wy = as.matrix(st_drop_geometry(pts[,1:3]))

( nn <- knn(pts, meuse, k=2, ids = "IDS", indexes = TRUE,
            weights.y=wy, weights.x=wx) )
  plot(st_geometry(pts), pch=19, main="KNN")
    plot(st_geometry(meuse[nn[,1],]), pch=19, col="red")

# Using coordinate matrices
y <- st_coordinates(pts)[,1:2]
x <- st_coordinates(meuse)[,1:2]
knn(y, x, k=2)

} else {
  cat("Please install sp package to run example", "\n")
}
```

`lai` *Leaf Area Index*

Description

Remote sensing measure of LAI (leaf area per ground-unit area)

Usage

```
lai(ndvi, method = c("Jonckheere", "Chen"))
```

Arguments

`ndvi`	NDVI in floating point standard scale range (-1 to 1)
`method`	Method to use for index options c("Jonckheere", "Chen")

Details

This function calculates the Leaf Area Index (LAI) representing the amount of leaf area per unit of ground area. This is an important parameter for understanding the structure and function of vegetation, as it affects processes such as photosynthesis, transpiration, and carbon cycling. These two approaches are based on the empirical relationship between NDVI and LAI, which has been observed in many studies, and it is a widely used method for estimating LAI from remote sensing data. The formulas are derived from the fact that vegetation with higher LAI tends to have higher reflectance in the near-infrared (NIR) band and lower reflectance in the red band, resulting in higher

NDVI values. But still, the exact relationship between NDVI and LAI can vary depending on factors such as vegetation type, canopy structure, and environmental conditions.

Value

A terra SpatRaster object with derived LAI vaues

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

References

Jonckheere, I., Fleck, S., Nackaerts, K., Muys, B., Coppin, P. (2004). A comparison of two methods to retrieve the leaf area index (LAI) from SPOT-4 HRVIR data. International Journal of Remote Sensing, 25(21):4407–4425.

Chen, J. M., Liu, R., & Ju, W. (2014). A simple and effective method for estimating leaf area index from Landsat imagery. Remote Sensing of Environment, 152:538–548.

Examples

```
library(terra)
lsat <- rast(system.file("/extdata/Landsat_TM5.tif", package="spatialEco"))
   plotRGB(lsat, r=3, g=2, b=1, scale=1.0, stretch="lin")

  ndvi <-  ( lsat[[4]] - lsat[[3]] ) / (lsat[[4]] + lsat[[3]])

 # Using Jonckheere et al., (2004) method
 lai01 <- lai(ndvi)
   plot(lai01)
```

`local.min.max` *Local minimum and maximum*

Description

Calculates the local minimums and maximums in a numeric vector, indicating inflection points in the distribution.

Usage

```
local.min.max(x, dev = mean, plot = TRUE, add.points = FALSE, ...)
```

Arguments

`x`	A numeric vector
`dev`	Deviation statistic (mean or median)
`plot`	plot the minimum and maximum values with the distribution (TRUE/FALSE)
`add.points`	Should all points of x be added to plot (TRUE/FALSE)
`...`	Arguments passed to plot

Details

Useful function for identifying inflection or enveloping points in a distribution

Value

A list object with:

- minima - minimum local values of x
- maxima - maximum local values of x
- mindev - Absolute deviation of minimum from specified deviation statistic (dev argument)
- maxdev - Absolute deviation of maximum from specified deviation statistic (dev argument)

Author(s)

Jeffrey S. Evans jeffrey_evans@tnc.org

Examples

```
x <- rnorm(100,mean=1500,sd=800)
( lmm <- local.min.max(x, dev=mean, add.points=TRUE,
                       main="Local Minima and Maxima") )

# return only local minimum values
   local.min.max(x)$minima
```

`loess.boot` *Loess Bootstrap*

Description

Bootstrap of a Local Polynomial Regression (loess)

Usage

```
loess.boot(x, y, nreps = 100, confidence = 0.95, ...)
```

Arguments

`x`	Independent variable
`y`	Dependent variable
`nreps`	Number of bootstrap replicates
`confidence`	Fraction of replicates contained in confidence region
`...`	Additional arguments passed to loess function

Details

The function fits a loess curve and then calculates a symmetric nonparametric bootstrap with a confidence region. Fitted curves are evaluated at a fixed number of equally-spaced x values, regardless of the number of x values in the data. Some replicates do not include the values at the lower and upper end of the range of x values. If the number of such replicates is too large, it becomes impossible to construct a confidence region that includes a fraction "confidence" of the bootstrap replicates. In such cases, the left and/or right portion of the confidence region is truncated.

Value

list object containing

- nreps Number of bootstrap replicates
- confidence Confidence interval (region)
- span alpha (span) parameter used loess fit
- degree polynomial degree used in loess fit
- normalize Normalized data (TRUE/FALSE)
- family Family of statistic used in fit
- parametric Parametric approximation (TRUE/FALSE)
- surface Surface fit, see loess.control
- data data.frame of x,y used in model
- fit data.frame including:
  1. x - Equally-spaced x index (see NOTES)
  2. y.fit - loess fit
  3. up.lim - Upper confidence interval
  4. low.lim - Lower confidence interval
  5. stddev - Standard deviation of loess fit at each x value

Author(s)

Jeffrey S. Evans jeffrey_evans@tnc.org

References

Cleveland, WS, (1979) Robust Locally Weighted Regression and Smoothing Plots Journal of the American Statistical Association 74:829-836

Efron, B., and R. Tibshirani (1993) An Introduction to the Bootstrap Chapman and Hall, New York

Hardle, W., (1989) Applied Nonparametric Regression Cambridge University Press, NY.

Tibshirani, R. (1988) Variance stabilization and the bootstrap. Biometrika 75(3):433-44.

Examples

```
n=1000
x <- seq(0, 4, length.out=n)
y <- sin(2*x)+ 0.5*x + rnorm(n, sd=0.5)
sb <- loess.boot(x, y, nreps=99, confidence=0.90, span=0.40)
plot(sb)
```

`loess.ci` *Loess with confidence intervals*

Description

Calculates a local polynomial regression fit with associated confidence intervals

Usage

```
loess.ci(y, x, p = 0.95, plot = FALSE, ...)
```

Arguments

`y`	Dependent variable, vector
`x`	Independent variable, vector
`p`	Percent confidence intervals (default is 0.95)
`plot`	Plot the fit and confidence intervals
`...`	Arguments passed to loess

Value

A list object with:

- loess Predicted values
- se Estimated standard error for each predicted value
- lci Lower confidence interval
- uci Upper confidence interval
- df Estimated degrees of freedom
- rs Residual scale of residuals used in computing the standard errors

Author(s)

Jeffrey S. Evans jeffrey_evans@tnc.org

References

W. S. Cleveland, E. Grosse and W. M. Shyu (1992) Local regression models. Chapter 8 of Statistical Models in S eds J.M. Chambers and T.J. Hastie, Wadsworth & Brooks/Cole.

Examples

```
 x <- seq(-20, 20, 0.1)
 y <- sin(x)/x + rnorm(length(x), sd=0.03)
 p <- which(y == "NaN")
   y <- y[-p]
   x <- x[-p]

opar <- par(no.readonly=TRUE)
  par(mfrow=c(2,2))
    lci <- loess.ci(y, x, plot=TRUE, span=0.10)
    lci <- loess.ci(y, x, plot=TRUE, span=0.30)
    lci <- loess.ci(y, x, plot=TRUE, span=0.50)
    lci <- loess.ci(y, x, plot=TRUE, span=0.80)
par(opar)
```

`logistic.regression` *Logistic and Auto-logistic regression*

Description

Performs a logistic (binomial) or auto-logistic (spatially lagged binomial) regression using maximum likelihood or penalized maximum likelihood estimation.

Usage

```
logistic.regression(
  ldata,
  y,
  x,
  penalty = TRUE,
  autologistic = FALSE,
  coords = NULL,
  bw = NULL,
  type = "inverse",
  style = "W",
  longlat = FALSE,
  ...
)
```

Arguments

`ldata`	data.frame object containing variables
`y`	Dependent variable (y) in ldata
`x`	Independent variable(s) (x) in ldata
`penalty`	Apply regression penalty (TRUE/FALSE)
`autologistic`	Add auto-logistic term (TRUE/FALSE)
`coords`	Geographic coordinates for auto-logistic model matrix or sp object.
`bw`	Distance bandwidth to calculate spatial lags (if empty neighbors result, need to increase bandwidth). If not provided it will be calculated automatically based on the minimum distance that includes at least one neighbor.
`type`	Neighbor weighting scheme (see autocov_dist)
`style`	Type of neighbor matrix (Wij), default is mean of neighbors
`longlat`	Are coordinates (coords) in geographic, lat/long (TRUE/FALSE)
`...`	Additional arguments passed to lrm

Details

It should be noted that the auto-logistic model (Besag 1972) is intended for exploratory analysis of spatial effects. Auto-logistic are know to underestimate the effect of environmental variables and tend to be unreliable (Dormann 2007). Wij matrix options under style argument - B is the basic binary coding, W is row standardized (sums over all links to n), C is globally standardized (sums over all links to n), U is equal to C divided by the number of neighbours (sums over all links to unity) and S is variance-stabilizing. Spatially lagged y defined as: W(y)ij=sumj_(Wij yj)/ sumj_(Wij) where; Wij=1/Euclidean(i,j) If the object passed to the function is an sp class there is no need to call the data slot directly via "object@data", just pass the object name.

Value

A list class object with the following components:

- model - lrm model object (rms class)
- bandwidth - If AutoCov = TRUE returns the distance bandwidth used for the auto-covariance function
- diagTable - data.frame of regression diagnostics
- coefTable - data.frame of regression coefficients (log-odds)
- Residuals - data.frame of residuals and standardized residuals
- AutoCov - If an auto-logistic model, AutoCov represents lagged auto-covariance term

Author(s)

Jeffrey S. Evans jeffrey_evans@tnc.org

References

Besag, J.E., (1972) Nearest-neighbour systems and the auto-logistic model for binary data. Journal of the Royal Statistical Society, Series B Methodological 34:75-83

Dormann, C.F., (2007) Assessing the validity of autologistic regression. Ecological Modelling 207:234-242

Le Cessie, S., Van Houwelingen, J.C., (1992) Ridge estimators in logistic regression. Applied Statistics 41:191-201

Shao, J., (1993) Linear model selection by cross-validation. JASA 88:486-494

See Also

`lrm`

`autocov_dist`

Examples

```
p = c("sf", "sp", "spdep", "rms")
if(any(!unlist(lapply(p, requireNamespace, quietly=TRUE)))) {
  m = which(!unlist(lapply(p, requireNamespace, quietly=TRUE)))
  message("Can't run examples, please install ", paste(p[m], collapse = " "))
} else {
  invisible(lapply(p, require, character.only=TRUE))

data(meuse, package = "sp")
meuse <- st_as_sf(meuse, coords = c("x", "y"), crs = 28992,
                  agr = "constant")
  meuse$DepVar <- rbinom(nrow(meuse), 1, 0.5)

#### Logistic model
lmodel <- logistic.regression(meuse, y='DepVar',
                  x=c('dist','cadmium','copper'))
  lmodel$model
    lmodel$diagTable
      lmodel$coefTable

#### Logistic model with factorial variable
lmodel <- logistic.regression(meuse, y='DepVar',
            x=c('dist','cadmium','copper', 'soil'))
  lmodel$model
    lmodel$diagTable
      lmodel$coefTable

 ### Auto-logistic model using 'autocov_dist' in 'spdep' package
 lmodel <- logistic.regression(meuse, y='DepVar',
             x=c('dist','cadmium','copper'), autologistic=TRUE,
             coords=st_coordinates(meuse), bw=5000)
   lmodel$model
     lmodel$diagTable
       lmodel$coefTable
   est <- predict(lmodel$model, type='fitted.ind')
```

```
 #### Add residuals, standardized residuals and estimated probabilities
 VarNames <- rownames(lmodel$model$var)[-1]
   meuse$AutoCov <- lmodel$AutoCov
   meuse$Residual <- lmodel$Residuals[,1]
   meuse$StdResid <- lmodel$Residuals[,2]
   meuse$Probs <- predict(lmodel$model,
                          sf::st_drop_geometry(meuse[,VarNames]),
                       type='fitted')

#### Plot fit and probabilities

resid(lmodel$model, "partial", pl="loess")

# plot residuals
resid(lmodel$model, "partial", pl=TRUE)

# global test of goodness of fit
resid(lmodel$model, "gof")

# Approx. leave-out linear predictors
lp1 <- resid(lmodel$model, "lp1")

# Approx leave-out-1 deviance
-2 * sum(meuse$DepVar * lp1 + log(1-plogis(lp1)))

# plot estimated probabilities at points
plot(meuse['Probs'], pch=20)

}
```

`max_extent` *Maximum extent of multiple rasters*

Description

returns a extent polygon representing maximum extent of input rasters

Usage

```
max_extent(x, ...)
```

Arguments

`x`	terra SpatRaster class object
`...`	additional SpatRaster class objects in same projection

Details

Creates a maximum extent polygon of all specified rasters

Value

An sf POLYGON class object representing maximum extents

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
library(terra)

r1 <- rast(ext(61.87125, 76.64458, 23.90153, 37.27042))
r2 <- rast(ext(67.66625, 81.56847, 20.38458, 35.67347))
r3 <- rast(ext(72.64792,84.38125,5.91125,28.13347 ))

( e <- max_extent(r1, r2, r3) )
plot(e, border=NA)
  plot(ext(r1), border="red", add=TRUE)
  plot(ext(r2), border="green", add=TRUE)
  plot(ext(r3), border="blue", add=TRUE)
  plot(e, border="black", add=TRUE)

 sf::st_bbox(e) # full extent
```

`mean_angle` *Mean Angle*

Description

Calculates the mean angle of a vector

Usage

```
mean_angle(a, angle = c("degree", "radians"))
```

Arguments

`a`	vector of angle values
`angle`	("degree", "radians") to define angle in degrees or radians

Details

The arithmetic mean is not correct for calculating the central tendency of angles. This function is intended to return the mean angle for slope or aspect, which could be used in a focal or zonal function.

Value

A vector of mean angle

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
library(terra)
mean_angle(c(180, 10))
  mean(c(180, 10))
mean_angle(c(90, 180, 70, 60))
  mean(c(90, 180, 70, 60))
mean_angle(c(90, 180, 270, 360))
  mean(c(90, 180, 270, 360))

elev <- rast(system.file("extdata/elev.tif", package="spatialEco"))
asp <- terrain(elev, v="aspect")
s <- buffer(spatSample(asp, 20, as.points=TRUE,
            na.rm=TRUE, values=FALSE), 5000)

plot(asp)
  plot(s, add=TRUE)

d <- extract(asp, s)
cat("Mean angles of aspect", "\n")
  tapply(d[,2], d[,1], mean_angle)
cat("arithmetic means of aspect", "\n")
  tapply(d[,2], d[,1], mean, na.rm=TRUE)
```

`moments` *moments*

Description

Calculate statistical moments of a distribution

Usage

```
moments(x, plot = FALSE)
```

Arguments

`x`	numeric vector
`plot`	plot of distribution (TRUE/FALSE)

Value

A vector with the following values:

- min Minimum
- 25th 25th percentile
- mean Arithmetic mean
- gmean Geometric mean
- hmean Harmonic mean
- median 50th percentile
- 7th5 75th percentile
- max Maximum
- stdv Standard deviation
- var Variance
- cv Coefficient of variation (percent)
- mad Median absolute deviation
- skew Skewness
- kurt Kurtosis
- nmodes Number of modes
- mode Mode (dominate)

Author(s)

Jeffrey S. Evans jeffrey_evans@tnc.org

Examples

```
x <- runif(1000,0,100)
( d <- moments(x, plot=TRUE) )
( mode.x <- moments(x, plot=FALSE)[16] )
```

`morans.plot` *Autocorrelation Plot*

Description

Autocorrelation plot (Anselin 1996), following Chen (2015), aka, Moran's-I plot (univariate or bivariate)

Usage

```
morans.plot(
  x,
  y = NULL,
  coords = NULL,
  type.ac = c("xy", "yx"),
  dist.function = "inv.power",
  scale.xy = TRUE,
  scale.morans = FALSE,
  ...
)
```

Arguments

`x`	Vector of x response variables
`y`	Vector of y response variables
`coords`	A matrix of coordinates corresponding to [x,y]
`type.ac`	Type of autocorrelation plot ("xy", "yx")
`dist.function`	("inv.power", "neg.exponent")
`scale.xy`	(TRUE/FALSE) scale the x,y vectors
`scale.morans`	(FALSE/TRUE) standardize the Moran's index to an expected [-1 to 1]?
`...`	Additional arguments passed to plot

Details

The argument "type" controls the plot for x influencing y (type="xy") or y influencing x (type="yx"). If y is not defined then the statistic is univariate and only the "xy" plot will be available. The linear relationship between x and its spatial lag (Wx) is indicative of the spatial autoregressive process, underlying the spatial dependence. The statistic can be autocorrelation (univariate or cross-correlation (bivariate). The quadrants are the zero intercept for random autocorrelation and the red line represents the trend in autocorrelation. The quadrants in the plot indicate the type of spatial association/interaction (Anselin 1996). For example the upper-left quadrant represents negative associations of low values surrounded by high and the lower-right quadrant represents negative associations of high values surrounded by low.

If y is not specified the univariate statistic for x is returned. the coords argument is only used if k = NULL. Can also be an sp object with relevant x,y coordinate slot (ie., points or polygons). If w = NULL, the default method for deriving spatial weights matrix, options are: inverse power or negative exponent. If scale.xy = FALSE it is assumed that they are already scaled following Chen (2015).

Value

A plot of the scaled variable against its spatially lagged values.

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

References

Chen., Y. (2015) A New Methodology of Spatial Cross-Correlation Analysis. PLoS One 10(5):e0126158. doi:10.1371/journal.pone.0126158

Anselin, L. (1996) The Moran scatterplot as an ESDA tool to assess local instability in spatial association. pp. 111-125 in M. M. Fischer, H. J. Scholten and D. Unwin (eds) Spatial analytical perspectives on GIS, London, Taylor and Francis

Anselin, L. (1995) Local indicators of spatial association, Geographical Analysis, 27:93-115

Examples

```
 p = c("sf", "sp", "spdep")
 if(any(!unlist(lapply(p, requireNamespace, quietly=TRUE)))) {
   m = which(!unlist(lapply(p, requireNamespace, quietly=TRUE)))
   message("Can't run examples, please install ", paste(p[m], collapse = " "))
 } else {
 invisible(lapply(p, require, character.only=TRUE))

 data(meuse, package = "sp")
 meuse <- st_as_sf(meuse, coords = c("x", "y"), crs = 28992, agr = "constant")

# Autocorrelation (univariate)
morans.plot(meuse$zinc, coords = st_coordinates(meuse)[,1:2])

# Cross-correlation of: x influencing y and y influencing x
opar <- par(no.readonly=TRUE)
  par(mfrow=c(1,2))
    morans.plot(x=meuse$zinc, y=meuse$copper, coords = st_coordinates(meuse)[,1:2],
                scale.morans = TRUE)
    morans.plot(x=meuse$zinc, y=meuse$copper, coords = st_coordinates(meuse)[,1:2],
                scale.morans = TRUE, type.ac="yx")
par(opar)
}
```

`nni` *Average Nearest Neighbor Index (NNI)*

Description

Calculates the NNI as a measure of clustering or dispersal

Usage

```
nni(x, win = c("hull", "extent"))
```

Arguments

`x`	An sf point object
`win`	Type of window 'hull' or 'extent'

Details

The nearest neighbor index is expressed as the ratio of the observed distance divided by the expected distance. The expected distance is the average distance between neighbors in a hypothetical random distribution. If the index is less than 1, the pattern exhibits clustering; if the index is greater than 1, the trend is toward dispersion or competition. The Nearest Neighbor Index is calculated as:

- Mean Nearest Neighbor Distance (observed) D(nn) = sum(min(Dij)/N)
- Mean Random Distance (expected) D(e) = 0.5 SQRT(A/N)
- Nearest Neighbor Index NNI = D(nn)/D(e) Where; D=neighbor distance, A=Area

Value

list object containing NNI = nearest neighbor index, z.score = Z Score value, p = p value, expected.mean.distance = Expected mean distance, observed.mean.distance = Observed meand distance.

Author(s)

Jeffrey S. Evans jeffrey_evans@tnc.org

References

Clark, P.J., and F.C. Evans (1954) Distance to nearest neighbour as a measure of spatial relationships in populations. Ecology 35:445-453

Cressie, N (1991) Statistics for spatial data. Wiley & Sons, New York.

Examples

```
p = c("sf", "sp")
  if(any(!unlist(lapply(p, requireNamespace, quietly=TRUE)))) {
    m = which(!unlist(lapply(p, requireNamespace, quietly=TRUE)))
    message("Can't run examples, please install ", paste(p[m], collapse = " "))
  } else {
  invisible(lapply(p, require, character.only=TRUE))

  data(meuse, package = "sp")
  meuse <- sf::st_as_sf(meuse, coords = c("x", "y"),
                        crs = 28992, agr = "constant")
  nni(meuse)
  }
```

`nth.values` *Nth values*

Description

Returns the Nth highest or lowest values in a vector

Usage

```
nth.values(x, N = 2, smallest = FALSE)
```

Arguments

`x`	Numeric vector
`N`	Number of (Nth) values returned
`smallest`	(FALSE/TRUE) Return the highest, else smallest values

Details

This function returns n lowest or highest elements in a vector

Value

Numeric vector of Nth values

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
nth.values(1:20, N=3, smallest = TRUE)
nth.values(1:20, N=3)
```

`o.ring` *Inhomogeneous O-ring*

Description

Calculates the inhomogeneous O-ring point pattern statistic (Wiegand & Maloney 2004)

Usage

```
o.ring(x, inhomogeneous = FALSE, ...)
```

Arguments

`x`	spatstat ppp object
`inhomogeneous`	(FALSE/TRUE) Run homogeneous (pcf) or inhomogeneous (pcfinhom)
`...`	additional arguments passed to pcf or pcfinhom

Details

The function K(r) is the expected number of points in a circle of radius r centered at an arbitrary point (which is not counted), divided by the intensity l of the pattern. The alternative pair correlation function g(r), which arises if the circles of Ripley's K-function are replaced by rings, gives the expected number of points at distance r from an arbitrary point, divided by the intensity of the pattern. Of special interest is to determine whether a pattern is random, clumped, or regular.

Using rings instead of circles has the advantage that one can isolate specific distance classes, whereas the cumulative K-function confounds effects at larger distances with effects at shorter distances. Note that the K-function and the O-ring statistic respond to slightly different biological questions. The accumulative K-function can detect aggregation or dispersion up to a given distance r and is therefore appropriate if the process in question (e.g., the negative effect of competition) may work only up to a certain distance, whereas the O-ring statistic can detect aggregation or dispersion at a given distance r. The O-ring statistic has the additional advantage that it is a probability density function (or a conditioned probability spectrum) with the interpretation of a neighborhood density, which is more intuitive than an accumulative measure.

Value

plot of o-ring and data.frame with plot labels and descriptions

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

References

Wiegand T., and K. A. Moloney (2004) Rings, circles and null-models for point pattern analysis in ecology. Oikos 104:209-229

Examples

```
if (require(spatstat.explore, quietly = TRUE)) {
data(lansing)
  x  <- spatstat.geom::unmark(split(lansing)$maple)
  o.ring(x)

} else {
  cat("Please install spatstat.explore package to run example", "\n")
}
```

`oli.asw` *Query AWS-OLI*

Description

Query of Amazon AWS OLI-Landsat 8 cloud service

Usage

```
oli.asw(path, row, dates, cloud.cover = 10, processing)
```

Arguments

`path`	landsat path
`row`	landsat row
`dates`	dates, single or start-stop range in YYYY-MM-DD format
`cloud.cover`	percent cloud cover
`processing`	processing level ("L1GT" or "L1T")

Details

Amazons AWS cloud service is hosting OLI Landsat 8 data granules `https://registry.opendata.aws/landsat-8` `https://aws.amazon.com/blogs/aws/start-using-landsat-on-aws/`

USGS Landsat collections: `https://www.usgs.gov/landsat-missions` Pre-collection processing levels: "L1T", "L1GT", "L1G" Collection 1 processing levels: "L1TP", "L1GT", "L1GS" "L1T" and "L1TP" - Radiomertically calibrated and orthorectified (highest level processing) "L1GT" and "L1GT" - Radiomertically calibrated and systematic geometric corrections "L1G" and "L1GS" - Radiomertically calibrated with systematic ephemeris correction

Value

data.frame object with:

- entityId - Granule ID
- L = Landsat
- X = Sensor
- SS = Satellite
- PPP = WRS path
- RRR = WRS row
- YYYYMMDD = Acquisition date
- yyyymmdd = Processing date
- CC = Collection number
- TX = Collection category

- acquisitionDate - POSIXct YYYY-MM-DD (eg., 2015-01-02)
- cloudCover -
- processingLevel - USGS processing level
- path - Landsat path
- row - Landsat row

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
# Query path 126, row 59, 2013-04-15 to 2017-03-09, <20% cloud cover
( p126r59.oli <- oli.asw(path=126, row=59, dates = c("2013-04-15", "2017-03-09"),
                           cloud.cover = 20) )

# Download images from query
 bands <- c("_B1.TIF", "_B2.TIF", "_B3.TIF", "_B4.TIF", "_B5.TIF",
            "_B6.TIF","_B7.TIF", "_B8.TIF", "_B9.TIF", "_B10.TIF",
         "_B11.TIF", "_BQA.TIF","_MTL.txt")
  for(i in 1:length(p126r59.oli$download_url)) {
    oli.url <- gsub("/index.html","",p126r59.oli$download_url[i])
 all.bands <- paste(oli.url, paste0(unlist(strsplit(oli.url, "/"))[8], bands), sep="/")
   for(j in all.bands) {
        try(utils::download.file(url=j, destfile=basename(j), mode = "wb"))
       }
  }
```

`optimal.k` *optimalK*

Description

Find optimal k of k-Medoid partitions using silhouette widths

Usage

```
optimal.k(x, nk = 10, plot = TRUE, cluster = TRUE, clara = FALSE, ...)
```

Arguments

`x`	Numeric dataframe, matrix or vector
`nk`	Number of clusters to test (2:nk)
`plot`	(TRUE / FALSE) Plot cluster silhouettes(TRUE/FALSE)

`cluster`	(TRUE / FALSE) Create cluster object with optimal k
`clara`	(FALSE / TRUE) Use clara model for large data
`...`	Additional arguments passed to clara

Value

Object of class clust "pam" or "clara" with tested silhouette values

Author(s)

Jeffrey S. Evans <jeffrey_evans<at>tnc.org>

References

Theodoridis, S. & K. Koutroumbas(2006) Pattern Recognition 3rd ed.

See Also

`pam` for details on Partitioning Around Medoids (PAM)

`clara` for details on Clustering Large Applications (clara)

Examples

```
if (require(cluster, quietly = TRUE)) {
  x <- rbind(cbind(rnorm(10,0,0.5), rnorm(10,0,0.5)),
             cbind(rnorm(15,5,0.5), rnorm(15,5,0.5)))

  clust <- optimal.k(x, 20, plot=TRUE, cluster=TRUE)
    plot(silhouette(clust$model), col = c('red', 'green'))
      plot(clust$model, which.plots=1, main='K-Medoid fit')

# Extract multivariate and univariate mediods (class centers)
  clust$model$medoids
    pam(x[,1], 1)$medoids

# join clusters to data
  x <- data.frame(x, k=clust$model$clustering)

} else {
  cat("Please install cluster package to run example", "\n")
}
```

`optimized.sample.variance`

Optimized sample variance

Description

Draws an optimal sample that minimizes or maximizes the sample variance

Usage

```
optimized.sample.variance(x, n, type = "maximized")
```

Arguments

`x`	A vector to draw a sample from
`n`	Number of samples to draw
`type`	Type of sample variance optimization c("maximized", "minimized")

Value

A data.frame with "idx" representing the index of the original vector and "y" is the value of the sampled data

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
library(sf)
if (require(sp, quietly = TRUE)) {
  data(meuse, package = "sp")
  meuse <- st_as_sf(meuse, coords = c("x", "y"), crs = 28992, agr = "constant")

 n = 15
 # Draw n samples that maximize the variance of y
 ( max.sv <- optimized.sample.variance(meuse$zinc, 15) )

 # Draw n samples that minimize the variance of y
 ( min.sv <- optimized.sample.variance(meuse$zinc, 15,
              type="minimized") )

 # Plot results
 plot(st_geometry(meuse), pch=19, col="grey")
   plot(st_geometry(meuse[max.sv$idx,]), col="red", add=TRUE, pch=19)
     plot(st_geometry(meuse[min.sv$idx,]), col="blue", add=TRUE, pch=19)
    box()
     legend("topleft", legend=c("population","maximized variance",
```

```
            "minimized variance"), col=c("grey","red","blue"),
            pch=c(19,19,19))

} else {
  cat("Please install sp package to run example", "\n")
}
```

---

outliers *Outliers*

Description

Identify outliers using modified Z-score

Usage

```
outliers(x, s = 1.4826)
```

Arguments

`x`	A numeric vector
`s`	Scaling factor for mad statistic

Value

value for the modified Z-score

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

References

Iglewicz, B. & D.C. Hoaglin (1993) How to Detect and Handle Outliers, American Society for Quality Control, Milwaukee, WI.

Examples

```
# Create data with 3 outliers
   x <- seq(0.1, 5, length=100)
   x[98:100] <- c(100, 55, 250)

# Calculate Z score
    Z <- outliers(x)

# Show number of extreme outliers using Z-score
    length(Z[Z > 9.9])

# Remove extreme outliers
    x <- x[-which(Z > 9.9)]
```

`overlap` *Niche overlap (Warren's-I)*

Description

Similarity Statistic for Quantifying Niche Overlap using Warren's-I

Usage

```
overlap(x, y)
```

Arguments

`x`	A matrix or SpatRaster raster class object
`y`	A matrix or SpatRaster raster class object with the same dimensions of x

Details

The overlap function computes the I similarity statistic (Warren et al. 2008) of two overlapping niche estimates. Similarity is based on the Hellenger distance. It is assumed that the input data share the same extent and cellsize and all values are positive.

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). The function is based on code from Jeremy VanDerWal

Value

A vector (single value) representing the I similarity statistic

Author(s)

Jeffrey Evans <jeffrey_evans@tnc.org> and Jeremy VanDerWal

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

```
# add degree of separation in two matrices
p1 <- abs(matrix(1:50,nr=50,nc=50) +
         runif(n = 2500, min = -1, max = 1))
p2 <- abs(matrix(1:50,nr=50,nc=50) +
         rnorm(n = 2500, mean = 1, sd = 1))
```

```
# High overlap/similarity
( I <- overlap(p1,p2) )
```

`parea.sample` *Percent area sample*

Description

Creates a point sample of polygons where n is based on percent area

Usage

```
parea.sample(x, pct = 0.1, join = FALSE, sf = 4046.86, stype = "random", ...)
```

Arguments

`x`	An sf POLYGON object
`pct`	Percent of area sampled
`join`	FALSE/TRUE Join polygon attributed to point sample
`sf`	Scaling factor (default is meters to acres conversion factor)
`stype`	Sampling type ('random', 'regular', 'nonaligned', 'hexagonal')
`...`	Additional arguments passed to spsample

Details

This function results in an adaptive sample based on the area of each polygon. The default scaling factor (sf) converts meters to acres. You can set sf=1 to stay in the native projection units

Value

An sf POINT object

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
library(sf)
nc <- st_read(system.file("shape/nc.shp", package="sf"))
  nc <- suppressWarnings(st_cast(nc[c(10,100),], "POLYGON"))

 ( ars <- parea.sample(nc, pct=0.001, join = TRUE, stype='random') )
     plot(st_geometry(nc))
        plot(st_geometry(ars), pch=19, add=TRUE)
```

`parse.bits` *Parse bits*

Description

Returns specified bit value based on integer input

Data such as MODIS the QC band are stored in bits. This function returns the value(s) for specified bit. For example, the MODIS QC flag are bits 0-1 with the bit value 00 representing the "LST produced, good quality" flag. When exported from HDF the QC bands are often in an 8 bit integer range (0-255). With this function you can parse the values for each bit to assign the flag values.

Usage

```
parse.bits(x, bit, depth = 8, order = c("reverse", "none"))
```

Arguments

`x`	Integer value
`bit`	A single or vector of bits to return
`depth`	The depth (length) of the bit range, default is 8
`order`	c("reverse", "none") sort order for the bits

Value

a vector or data.frame of parsed interger value(s) associated with input bit

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
# Return value for bit 5 for integer value 100
parse.bits(100, 5)

# Return value(s) for bits 0 and 1 for integer value 100
parse.bits(100, c(0,1))

# Return value(s) for bits 0 and 1 for integer values 0-255
for(i in 0:255) { print(parse.bits(i, c(0,1))) }


#### Applied Example using Harmonized Landsat Sentinel-2 QC

# Create dummy data and qc band
library(terra)
r <- rast(nrow=100, ncol=100)
  r[] <- round(runif(ncell(r), 0,1))
```

```
qc <- rast(nrow=100, ncol=100)
  qc[] <- round(runif(ncell(qc), 64,234))

# Calculate bit values from QC table
( qc_bits <- data.frame(int=0:255,
cloud = unlist(lapply(0:255, FUN=parse.bits, bit=1)),
shadow = unlist(lapply(0:255, FUN=parse.bits, bit=3)),
acloud = unlist(lapply(0:255, FUN=parse.bits, bit=2)),
cirrus = unlist(lapply(0:255, FUN=parse.bits, bit=0)),
aerosol = unlist(lapply(0:255, FUN=parse.bits, bit=c(7,6)))) )

# Query the results to create a vector of integer values indicating what to mask
#  cloud is bit 1 and shadow bit 3
m <- sort(unique(qc_bits[c(which(qc_bits$cloud == 1),
                           which(qc_bits$shadow == 1)
   ),]$int))

# Apply queried integer values to mask image with QA band
qc[qc %in% m] <- NA
r <- mask(r, qc)
```

`partial.cor` *Partial and Semi-partial correlation*

Description

Calculates a partial or semi-partial correlation with parametric and nonparametric options

Usage

```
partial.cor(
  x,
  y,
  z,
  method = c("partial", "semipartial"),
  statistic = c("kendall", "pearson", "spearman")
)
```

Arguments

`x`	A vector, data.frame or matrix with 3 columns
`y`	A vector same length as x
`z`	A vector same length as x
`method`	Type of correlation: "partial" or "semipartial"
`statistic`	Correlation statistic, options are: "kendall", "pearson", "spearman"

Details

Partial and semipartial correlations show the association between two variables when one or more peripheral variables are controlled to hold them constant.

Suppose we have three variables, X, Y, and Z. Partial correlation holds constant one variable when computing the relations two others. Suppose we want to know the correlation between X and Y holding Z constant for both X and Y. That would be the partial correlation between X and Y controlling for Z. Semipartial correlation holds Z constant for either X or Y, but not both, so if we wanted to control X for Z, we could compute the semipartial correlation between X and Y holding Z constant for X.

Value

data.frame containing:

- correlation - correlation coefficient
- p.value - p-value of correlation
- test.statistic - test statistic
- n - sample size
- Method - indicating partial or semipartial correlation
- Statistic - the correlation statistic used

Author(s)

Jeffrey S. Evans jeffrey_evans@tnc.org

Examples

```
air.flow = stackloss[,1]
water.temperature = stackloss[,2]
acid = stackloss[,3]

# Partial using Kendall (nonparametric) correlation
partial.cor(air.flow, water.temperature, acid)

scholar <- data.frame(
  HSGPA=c(3.0, 3.2, 2.8, 2.5, 3.2, 3.8, 3.9, 3.8, 3.5, 3.1),
 FGPA=c(2.8, 3.0, 2.8, 2.2, 3.3, 3.3, 3.5, 3.7, 3.4, 2.9),
  SATV =c(500, 550, 450, 400, 600, 650, 700, 550, 650, 550))

# Standard Pearson's correlations between HSGPA and FGPA
cor(scholar[,1], scholar[,2])

# Partial correlation using Pearson (parametric) between HSGPA
#   and FGPA, controlling for SATV
partial.cor(scholar, statistic="pearson")

# Semipartial using Pearson (parametric) correlation
partial.cor(x=scholar[,2], y=scholar[,1], z=scholar[,3],
            method="semipartial", statistic="pearson")
```

`plot.effect.size` *Plot effect size*

Description

Plot function for effect.size object

Usage

```
## S3 method for class 'effect.size'
plot(x, ...)
```

Arguments

`x`	A effect.size object
`...`	Additional arguments passed to plot

Value

Plot of effect size object with group effect sizes and 95

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

`plot.loess.boot` *Plot Loess Bootstrap*

Description

Plot function for loess.boot object

Usage

```
## S3 method for class 'loess.boot'
plot(x, ...)
```

Arguments

`x`	A loess.boot object
`...`	Additional arguments passed to plot

Value

plot of lowess bootstrap

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
n=1000
x <- seq(0, 4, length.out=n)
y <- sin(2*x)+ 0.5*x + rnorm(n, sd=0.5)
sb <- loess.boot(x, y, nreps = 99, confidence = 0.90, span = 0.40)
plot(sb)
```

`poly.regression` *Local Polynomial Regression*

Description

Calculates a Local Polynomial Regression for smoothing or imputation of missing data.

Usage

```
poly.regression(
  y,
  x = NULL,
  s = 0.75,
  impute = FALSE,
  na.only = FALSE,
  ci = FALSE,
  ...
)
```

Arguments

`y`	Vector to smooth or impute NA values
`x`	Optional x covariate data (must match dimensions of y)
`s`	Smoothing parameter (larger equates to more smoothing)
`impute`	(FALSE/TRUE) Should NA values be inputed
`na.only`	(FALSE/TRUE) Should only NA values be change in y
`ci`	(FALSE/TRUE) Should confidence intervals be returned
`...`	Additional arguments passed to loess

Details

This is a wrapper function for loess that simplifies data smoothing and imputation of missing values. The function allows for smoothing a vector, based on an index (derived automatically) or covariates. If the impute option is TRUE NA values are imputed, otherwise the returned vector will still have NA's present. If impute and na.only are both TRUE the vector is returned, without being smoothed but with imputed NA values filled in. The loess weight function is defined using the tri-cube weight function w(x) = (1-|x|^3)^3 where; x is the distance of a data point from the point the curve being fitted.

Value

If ci = FALSE, a vector of smoothed values, otherwise a list object with:

- loess - A vector, same length of y, representing the smoothed or inputed data
- lower.ci - Lower confidence interval
- upper.ci - Upper confidence interval

Author(s)

Jeffrey S. Evans jeffrey_evans@tnc.org

See Also

`loess` for loess ... model options

Examples

```
x <- seq(-20, 20, 0.1)
y <- sin(x)/x + rnorm(length(x), sd=0.03)
p <- which(y == "NaN")
y <- y[-p]
r <- poly.regression(y, ci=TRUE, s=0.30)

plot(y,type="l", lwd=0.5, main="s = 0.10")
  y.polygon <- c((r$lower.ci)[1:length(y)], (r$upper.ci)[rev(1:length(y))])
  x.polygon <- c(1:length(y), rev(1:length(y)))
  polygon(x.polygon, y.polygon, col="#00009933", border=NA)
     lines(r$loess, lwd=1.5, col="red")

# Impute NA values, replacing only NA's
y.na <- y
y.na[c(100,200,300)] <- NA
p.y <- poly.regression(y.na, s=0.10, impute = TRUE, na.only = TRUE)
y - p.y

plot(p.y,type="l", lwd=1.5, col="blue", main="s = 0.10")
  lines(y, lwd=1.5, col="red")
```

`polyPerimeter` *Polygon perimeter*

Description

Calculates the perimeter length(s) for a polygon object

Usage

```
polyPerimeter(x)
```

Arguments

`x`	sf POLYGON class object

Value

A vector of polygon perimeters in projection units

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
library(sf)
  polys <- st_read(system.file("shape/nc.shp", package="sf"))
    polys <- suppressWarnings(st_cast(polys[c(10,100),], "POLYGON"))

 polyPerimeter(polys)
```

`poly_trend` *Polynomial trend*

Description

Fits a polynomial trend using specified order

Usage

```
poly_trend(x, y, degree, ci = 0.95, plot = TRUE, ...)
```

Arguments

`x`	Vector of x
`y`	Vector of y
`degree`	Polynomial order (default 3)
`ci`	+/- confidence interval (default 0.95)
`plot`	Plot results (TRUE/FALSE)
`...`	Additional arguments passed to plot

Details

A fit using a lm(y ~ x + I(X^2) + I(X^3)) form will be correlated which, can cause problems. The function avoids undue correlation using orthogonal polynomials

Value

A poly.trend class (list) containing

- trend data.frame of fit polynomial and upper/lower confidence intervals
- model Class lm model object fit with poly term
- prameterCI Intercept confidence intervals of Nth order polynomials
- order Specified polynomial order

Author(s)

Jeffrey S. Evans jeffrey_evans@tnc.org

Examples

```
 set.seed(42)
 x <- seq(from=0, to=20, by=0.1)
 y <- (500 + 0.4 * (x-10)^3)
 noise <- y + rnorm(length(x), mean=10, sd=80)

 p <- poly_trend(x, noise, degree = 3, ci = 0.95,
                 main="3rd degree polynomial")

 dev.new(height=6, width=12)
   layout(matrix(c(1,2), 1, 2, byrow = TRUE))
   p <- poly_trend(x, noise, degree = 3,
                   main="3rd degree polynomial")
   p <- poly_trend(x, noise, degree = 6,
                   main="6th degree polynomial")

cat("Confidence intervals for", "1 -", p$order, "polynomials",  "\n")
  p$prameterCI
```

`pp.subsample` *Point process random subsample*

Description

Generates random subsample based on density estimate of observations

Usage

```
pp.subsample(
  x,
  n,
  window = "hull",
  sigma = "Scott",
  wts = NULL,
  gradient = 1,
  edge = FALSE
)
```

Arguments

`x`	An sf POINT class
`n`	Number of random samples to generate
`window`	Type of window (hull or extent)
`sigma`	Bandwidth selection method for KDE, default is 'Scott'. Options are 'Scott', 'Stoyan', 'Diggle', 'likelihood', and 'geometry'
`wts`	Optional vector of weights corresponding to point pattern
`gradient`	A scaling factor applied to the sigma parameter used to adjust the gradient decent of the density estimate. The default is 1, for no adjustment (downweight < 1 | upweight > 1)
`edge`	Apply Diggle edge correction (TRUE/FALSE)

Details

The window type creates a convex hull by default or, optionally, uses the maximum extent (envelope). The resulting bandwidth can vary widely by method. the 'diggle' method is intended for bandwidth representing 2nd order spatial variation whereas the 'scott' method will represent 1st order trend. the 'geometry' approach will also represent 1st order trend. for large datasets, caution should be used with the 2nd order 'likelihood' approach, as it is slow and computationally expensive. finally, the 'stoyan' method will produce very strong 2nd order results. '

Available bandwidth selection methods are:

- Scott - (Scott 1992), Scott's Rule for Bandwidth Selection (1st order)
- Diggle - (Berman & Diggle 1989), Minimise the mean-square error via cross validation (2nd order)

- likelihood - (Loader 1999), Maximum likelihood cross validation (2nd order)
- geometry - Bandwidth is based on simple window geometry (1st order)
- Stoyan - (Stoyan & Stoyan 1995), Based on pair-correlation function (strong 2nd order)
- User defined - using a numeric value for sigma

Value

sf class POINT geometry containing random subsamples

Author(s)

Jeffrey S. Evans jeffrey_evans@tnc.org

References

Berman, M. and Diggle, P. (1989) Estimating weighted integrals of the second-order intensity of a spatial point process. Journal of the Royal Statistical Society, series B 51, 81-92.

Fithian, W & T. Hastie (2013) Finite-sample equivalence in statistical models for presence-only data. Annals of Applied Statistics 7(4): 1917-1939

Hengl, T., H. Sierdsema, A. Radovic, and A. Dilo (2009) Spatial prediction of species distributions from occurrence-only records: combining point pattern analysis, ENFA and regression-kriging. Ecological Modelling, 220(24):3499-3511

Loader, C. (1999) Local Regression and Likelihood. Springer, New York.

Scott, D.W. (1992) Multivariate Density Estimation. Theory, Practice and Visualization. New York, Wiley.

Stoyan, D. and Stoyan, H. (1995) Fractals, random shapes and point fields: methods of geometrical statistics. John Wiley and Sons.

Warton, D.i., and L.C. Shepherd (2010) Poisson Point Process Models Solve the Pseudo-Absence Problem for Presence-only Data in Ecology. The Annals of Applied Statistics, 4(3):1383-1402

Examples

```
library(sf)
if(require(spatstat.explore, quietly = TRUE)) {
data(bei, package = "spatstat.data")

trees <- st_as_sf(bei)
  trees <- trees[-1,]

n=round(nrow(trees) * 0.10, digits=0)
trees.wrs <- pp.subsample(trees, n=n, window='hull')
  plot(st_geometry(trees), pch=19, col='black')
    plot(st_geometry(trees.wrs), pch=19, col='red', add=TRUE)
      box()
       title('10% subsample')
    legend('bottomright', legend=c('Original sample', 'Subsample'),
                 col=c('black','red'),pch=c(19,19))
```

```
} else {
  cat("Please install spatstat.explore package to run example", "\n")
}
```

`print.cross.cor` *Print spatial cross correlation*

Description

print method for class "cross.cor"

Usage

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

Arguments

`x`	Object of class cross.cor
`...`	Ignored

Value

When not simulated k=0, prints functions list object containing:

- I - Global autocorrelation statistic
- SCI - - A data.frame with two columns representing the xy and yx autocorrelation
- nsim - value of NULL to represent p values were derived from observed data (k=0)
- p - Probability based observations above/below confidence interval
- t.test - Probability based on t-test

When simulated (k>0), prints functions list object containing:

- I - Global autocorrelation statistic
- SCI - A data.frame with two columns representing the xy and yx autocorrelation
- nsim - value representing number of simulations
- global.p - p-value of global autocorrelation statistic
- local.p - Probability based simulated data using successful rejection of t-test
- range.p - Probability based on range of probabilities resulting from paired t-test

---

`print.effect.size` *Print effect size*

Description

print method for class "effect.size"

Usage

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

Arguments

`x`	Object of class effect.size
`...`	Ignored

Value

Prints the output data.frame contaning; effect size with upper and lower confidence and, mean and sd by group

---

`print.loess.boot` *Print Loess bootstrap model*

Description

print method for class "loess.boot"

Usage

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

Arguments

`x`	Object of class loess.boot
`...`	Ignored

Value

same as summary lowess.boot of data.frame including;

- nreps Number of bootstrap replicates
- confidence Confidence interval (region)
- span alpha (span) parameter used loess fit
- degree polynomial degree used in loess fit
- normalize Normalized data (TRUE/FALSE)
- family Family of statistic used in fit
- parametric Parametric approximation (TRUE/FALSE)
- surface Surface fit, see loess.control
- data data.frame of x,y used in model
- fit data.frame including:
  1. x - Equally-spaced x index
  2. y.fit - loess fit
  3. up.lim - Upper confidence interval
  4. low.lim - Lower confidence interval
  5. stddev - Standard deviation of loess fit at each x value

---

`print.poly.trend` *Print poly_trend*

Description

print method for class "poly.trend"

Usage

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

Arguments

`x`	Object of class poly.trend
`...`	Ignored

Value

Prints trend model summary, order and trend confidence intervals

---

`proximity.index` *Proximity Index*

Description

Calculates proximity index for a set of polygons

Usage

```
proximity.index(x, y = NULL, min.dist = 0, max.dist = 1000, background = NULL)
```

Arguments

`x`	A polygon class sp or sf object
`y`	Optional column in data containing classes
`min.dist`	Minimum threshold distance
`max.dist`	Maximum neighbor distance
`background`	Optional value in y column indicating background value

Value

A vector equal to nrow(x) of proximity index values, if a background value is specified NA values will be returned in the position(s) of the specified class

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

References

Gustafson, E.J., & G.R. Parker (1994) Using an Index of Habitat Patch Proximity for Landscape Design. Landscape and Urban Planning 29:117-130

Examples

```
library(sf)
if(require(sp, quietly = TRUE)) {
  data(meuse, package = "sp")
  meuse <- st_as_sf(meuse, coords = c("x", "y"), crs = 28992,
                   agr = "constant")
  meuse <- st_buffer(meuse, dist = meuse$elev * 5)
    meuse$LU <- sample(c("forest","nonforest"), nrow(meuse),
                      replace=TRUE)

# All polygon proximity index 1000 radius
( pidx <- proximity.index(meuse, min.dist = 1) )
  pidx[pidx > 1000] <- 1000
```

```
  # Class-level proximity index 1000 radius
  ( pidx.class <- proximity.index(meuse, y = "LU", min.dist = 1) )

  # plot index for all polygons
  meuse$pidx <- pidx
    plot(meuse["pidx"])

  # plot index for class-level polygons
  meuse$cpidx <- pidx.class
    plot(meuse["cpidx"])

  # plot index for just forest class
  forest <- meuse[meuse$LU == "forest",]
   plot(forest["cpidx"])

} else {
  cat("Please install sp package to run example", "\n")
}
```

`pseudo.absence` *Pseudo-absence random samples*

Description

Generates pseudo-absence samples based on density estimate of known locations

Usage

```
pseudo.absence(
  x,
  n,
  window = "hull",
  ref = NULL,
  s = NULL,
  sigma = "Scott",
  wts = NULL,
  KDE = FALSE,
  gradient = 1,
  p = NULL,
  edge = FALSE
)
```

Arguments

`x`	An sf POINT geometry object
`n`	Number of random samples to generate

`window`	Type of window (hull OR extent), overridden if mask provided
`ref`	Optional terra SpatRaster class raster. The resolution of the density estimate will match mask.
`s`	Optional resolution passed to window argument. Caution should be used due to long processing times associated with high resolution. In contrast, coarse resolution can exclude known points.
`sigma`	Bandwidth selection method for KDE, default is 'Scott'. Options are 'Scott', 'Stoyan', 'Diggle', 'likelihood', and 'geometry'
`wts`	Optional vector of weights corresponding to point pattern
`KDE`	Return KDE raster (TRUE/FALSE)
`gradient`	A scaling factor applied to the sigma parameter used to adjust the gradient decent of the density estimate. The default is 1, for no adjustment (downweight < 1 | upweight > 1)
`p`	Minimum value for probability distribution (must be > 0)
`edge`	Apply Diggle edge correction (TRUE/FALSE)

Details

The window type creates a convex hull by default or, optionally, uses the maximum extent (envelope). If a mask is provided the kde will represent areas defined by the mask and defines the area that pseudo absence data will be generated.

Available bandwidth selection methods are:

- Scott (Scott 1992), Scott's Rule for Bandwidth Selection (1st order)
- Diggle (Berman & Diggle 1989), Minimize the mean-square error via cross
- validation (2nd order)
- likelihood (Loader 1999), Maximum likelihood cross validation (2nd order)
- geometry, Bandwidth is based on simple window geometry (1st order)
- Stoyan (Stoyan & Stoyan 1995), Based on pair-correlation function (strong 2nd order)
- User defined numeric distance bandwidth

Value

A list class object with the following components:

- sample A sf POINT geometry object containing random samples
- kde A terra SpatRaster class of inverted Isotropic KDE estimates used as sample weights (IF KDE = TRUE)
- sigma Selected bandwidth of KDE

Note

resulting bandwidth can vary widely by method. the 'diggle' method is intended for selecting bandwidth representing 2nd order spatial variation whereas the 'scott' method will represent 1st order trend. the 'geometry' approach will also represent 1st order trend. For large datasets, caution should be used with the 2nd order 'likelihood' approach, as it is slow and computationally expensive. finally, the 'stoyan' method will produce very strong 2nd order results.

Author(s)

Jeffrey S. Evans jeffrey_evans@tnc.org

References

Berman, M. and Diggle, P. (1989) Estimating weighted integrals of the second-order intensity of a spatial point process. Journal of the Royal Statistical Society, series B 51, 81-92.

Fithian, W & T. Hastie (2013) Finite-sample equivalence in statistical models for presence-only data. Annals of Applied Statistics 7(4): 1917-1939

Hengl, T., H. Sierdsema, A. Radovic, and A. Dilo (2009) Spatial prediction of species distributions from occurrence-only records: combining point pattern analysis, ENFA and regression-kriging. Ecological Modelling, 220(24):3499-3511

Loader, C. (1999) Local Regression and Likelihood. Springer, New York.

Scott, D.W. (1992) Multivariate Density Estimation. Theory, Practice and Visualization. New York, Wiley.

Stoyan, D. and Stoyan, H. (1995) Fractals, random shapes and point fields: methods of geometrical statistics. John Wiley and Sons.

Warton, D.i., and L.C. Shepherd (2010) Poisson Point Process Models Solve the Pseudo-Absence Problem for Presence-only Data in Ecology. The Annals of Applied Statistics, 4(3):1383-1402

Examples

```
 p = c("sf", "sp", "terra", "spatstat.data")
 if(any(!unlist(lapply(p, requireNamespace, quietly=TRUE)))) {
   m = which(!unlist(lapply(p, requireNamespace, quietly=TRUE)))
   message("Can't run examples, please install ", paste(p[m], collapse = " "))
 } else {
   invisible(lapply(p, require, character.only=TRUE))

data(meuse, package = "sp")
meuse <- st_as_sf(meuse, coords = c("x", "y"), crs = 28992,
                  agr = "constant")

# Using a raster mask
r <- rast(ext(meuse), resolution=40, crs=crs(meuse))
  r[] <- rep(1,ncell(r))

pa <- pseudo.absence(meuse, n=100, window='hull', KDE=TRUE, ref = r,
                     sigma='Diggle', s=50)
  col.br <- colorRampPalette(c('blue','yellow'))
    plot(pa$kde, col=col.br(10))
      plot(st_geometry(meuse), pch=20, cex=1, add=TRUE)
        plot(st_geometry(pa$sample), col='red', pch=20, cex=1, add=TRUE)
          legend('top', legend=c('Presence', 'Pseudo-absence'),
                 pch=c(20,20),col=c('black','red'), bg="white")

# With clustered data
data(bei, package = "spatstat.data")
  trees <- st_as_sf(bei)
```

```
    trees <- trees[-1,]

trees.abs <- pseudo.absence(trees, n=100, window='extent', KDE=TRUE)
  col.br <- colorRampPalette(c('blue','yellow'))
    plot(trees.abs$kde, col=col.br(10))
     plot(st_geometry(trees), pch=20, cex=0.50, add=TRUE)
        plot(st_geometry(trees.abs$sample), col='red', pch=20, cex=1, add=TRUE)
          legend('top', legend=c('Presence', 'Pseudo-absence'),
                 pch=c(20,20),col=c('black','red'),bg=”white”)
}
```

---

pu *Biodiversity Planning Units*

Description

Subset of biodiversity planning units for Haiti ecoregional spatial reserve plan

Format

A sp SpatialPolygonsDataFrame with 5919 rows and 46 variables:

UNIT_ID Unique planning unit ID

DR_Dr_A Biodiversity target

DR_Dr_L Biodiversity target

Ht_Dr_A Biodiversity target

Ht_Dr_L Biodiversity target

DR_Ms_A Biodiversity target

DR_Ms_L Biodiversity target

Ht_Ms_L Biodiversity target

DR_LM_M Biodiversity target

H_LM_M_L Biodiversity target

H_LM_R_L Biodiversity target

DR_LM_R_L Biodiversity target

DR_Rn_L Biodiversity target

DR_LM_R_S Biodiversity target

DR_Rn_S Biodiversity target

DR_Ms_S Biodiversity target

Ht_Ms_A Biodiversity target

DR_Ms_E Biodiversity target

DR_Ms_I Biodiversity target

DR_Rn_E Biodiversity target

DR_Rn_I Biodiversity target

H_LM_R_E Biodiversity target

Ht_Ms_E Biodiversity target

Ht_Rn_E Biodiversity target

DR_Rn_A Biodiversity target

Ht_Rn_A Biodiversity target

Ht_Rn_I Biodiversity target

Ht_Dr_E Biodiversity target

Ht_Ms_S Biodiversity target

Ht_Dr_S Biodiversity target

Ht_Rn_L Biodiversity target

Ht_Th_A Biodiversity target

Ht_Th_L Biodiversity target

Ht_Th_S Biodiversity target

Ht_Dr_U Biodiversity target

Ht_Dr_I Biodiversity target

Ht_Ms_I Biodiversity target

H_LM_M_A Biodiversity target

H_LM_M_E Biodiversity target

H_LM_R_A Biodiversity target

H_LM_M_S Biodiversity target

H_LM_R_I Biodiversity target

H_LM_R_S Biodiversity target

Ht_Rn_S Biodiversity target

Ht_Ms_U Biodiversity target

Ht_Rn_U Biodiversity target

Source

"The Nature Conservancy"

References

Evans, J.S., S.R. Schill, G.T. Raber (2015) A Systematic Framework for Spatial Conservation Planning and Ecological Priority Design in St. Lucia, Eastern Caribbean. Chapter 26 in Central American Biodiversity : Conservation, Ecology and a Sustainable Future. F. Huettman (eds). Springer, NY.

quadrats *Quadrats*

Description

Creates quadrat polygons for sampling or analysis

Usage

```
quadrats(x, s = 250, n = 100, r = NULL, sp = FALSE)
```

Arguments

`x`	An sf POLYGONS object defining extent
`s`	Radius defining single or range of sizes of quadrats
`n`	Number of quadrats
`r`	A rotation factor for random rotation, default is NULL
`sp`	(FALSE | TRUE) Output sp class object

Details

The radius (s) parameter can be a single value or a range of values, representing a randomization range of resulting quadrat sizes. The rotation (r) parameter can also be used to defined a fixed rotation or random range of quadrat rotations. You can specify each of these parameters using an explicit vector that will be sampled eg., seq(100,300,0.5)

Value

an sf POLYGONS object with rotated polygon(s)

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
library(sf)
library(terra)

# read meuse data and create convex hull
if (require(sp, quietly = TRUE)) {
  data(meuse, package = "sp")
  meuse <- st_as_sf(meuse, coords = c("x", "y"), crs = 28992, agr = "constant")
  e <- st_convex_hull(st_union(meuse))

 # Fixed size 250 and no rotation
 s <- quadrats(e, s = 250, n = 10)
   plot(st_geometry(s))
```

```
# Variable sizes 100-300 and rotation of 0-45 degrees
s <- quadrats(e, s = c(100,300), n = 10, r = c(0,45))
  plot(st_geometry(s))

# Variable sizes 100-300 and no rotation
s <- quadrats(e, s = c(100,300), n = 10)
  plot(st_geometry(s))


} else {
  cat("Please install sp package to run example", "\n")
}
```

`random.raster` *Random raster*

Description

Create a random raster or raster stack using specified distribution

Usage

```
random.raster(
  r = NULL,
  n.row = 50,
  n.col = 50,
  n.layers = 1,
  x = seq(1, 10),
  min = 0,
  max = 1,
  mean = 0,
  sd = 1,
  p = 0.5,
  s = 1.5,
  mask = TRUE,
  distribution = c("random", "normal", "seq", "binominal", "gaussian")
)
```

Arguments

`r`	Optional existing terra raster defining nrow/ncol
`n.row`	Number of rows
`n.col`	Number of columns
`n.layers`	Number of layers in resulting raster stack

`x`	A vector of values to sample if distribution is "sample"
`min`	Minimum value of raster
`max`	Maximum value of raster
`mean`	Mean of centered distribution
`sd`	Standard deviation of centered distribution
`p`	p-value for binominal distribution
`s`	sigma value for Gaussian distribution
`mask`	(TRUE/FALSE) If r is provided, mask results to r
`distribution`	Available distributions, c("random", "normal", "seq", "binominal", "gaussian", "sample")

Details

Options for distributions are; random, normal, seq, binominal, gaussian and sample raster(s)

Value

terra SpatRaster object with random rasters

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
library(terra)

# Using existing raster to create random binominal
r <- rast(system.file("ex/elev.tif", package="terra"))
( rr <- random.raster(r, n.layers = 3, distribution="binominal") )
  plot(c(r,rr))

# default; random, nrows=50, ncols=50, n.layers=5
( rr <- random.raster() )

# specified; binominal, nrows=20, ncols=20, nlayers=5
( rr <- random.raster(n.layer=5, n.col=20, n.row=20,
                     distribution="binominal") )

# specified; gaussian, nrows=50, ncols=50, nlayers=1
( rr <- random.raster(n.col=50, n.row=50, s=8,
                      distribution="gaussian") )
   plot(rr)

# specified; sample, nrows=50, ncols=50, nlayers=1
( rr <- random.raster(n.layer=1, x=c(2,6,10,15),
                     distribution="sample" ) )
  freq(rr)
```

`raster.change` *Raster change between two nominal rasters*

Description

Compares two categorical rasters with a variety of statistical options

Usage

```
raster.change(
  x,
  y,
  s = 3,
  stat = c("kappa", "t.test", "cor", "entropy", "cross-entropy", "divergence"),
  ...
)
```

Arguments

`x`	A terra SpatRaster
`y`	A terra SpatRaster for comparison to x
`s`	Integer or matrix for defining Kernel, must be odd but not necessarily square
`stat`	Statistic to use in comparison, please see details for options.
`...`	Additional arguments passed to terra::focalPairs

Details

This function provides a various statistics for comparing two classified maps. Valid options are:

- kappa - Cohen's Kappa
- t.test - Two-tailed paired t-test
- cor - Persons Correlation
- entropy - Delta entropy
- cross-entropy - Cross-entropy loss function
- divergence - Kullback-Leibler divergence (relative entropy)

Kappa and t-test values < 0 are reported as 0. For a weighted kappa, a matrix must be provided that correspond to the pairwise weights for all values in both rasters. Delta entropy is derived by calculating Shannon's on each focal window then differencing them (e(x) - e(y)). The s argument can be a single scalar, defining a symmetrical kernel, two scalers defining the dimensions of the kernel eg., c(3,5) or a matrix defining the kernel say, resulting from terra::focalMat

Value

A terra SpatRaster layer containing one of the following layers:

- kappa - Kappa or Weighted Kappa statistic (if stat = "kappa")
- correlation - Paired t.test statistic (if stat = "cor")
- entropy - Local entropy (if stat = "entropy")
- divergence - Kullback-Leibler divergence (if stat = "divergence")
- cross.entropy - Local Cross-entropy (if stat = "cross.entropy")
- t.test - Paired t.test statistic (if stat = "t.test")
- p.value - p-value of the paired t.test statistic (if stat = "t.test")

Author(s)

Jeffrey S. Evans jeffrey_evans@tnc.org

References

Cohen, J. (1960). A coefficient of agreement for nominal scales. Educational and Psychological Measurement, 20:37-46

McHugh M.L. (2012) Interrater reliability: the kappa statistic. Biochemia medica, 22(3):276–282.

Kullback, S., R.A. Leibler (1951). On information and sufficiency. Annals of Mathematical Statistics. 22(1):79–86

Examples

```
library(sf)
library(terra)

e <- ext(179407.8, 181087.9, 331134.4, 332332.1)
r1 <- rast(e, resolution=20)
  r1[] <- sample(1:5, ncell(r1), replace=TRUE)
r2 <- rast(e, resolution=20)
  r2[] <- sample(1:5, ncell(r2), replace=TRUE)

d = 5 # kernel
( r.kappa <- raster.change(r1, r2, s = d) )
( r.ttest <- raster.change(r1, r2, s = d, stat="t.test") )
( r.ent <- raster.change(r1, r2, s = d, stat="entropy") )
( r.cor <- raster.change(r1, r2, s = d, stat="cor") )
( r.ce <- raster.change(r1, r2, s = d, stat = "cross-entropy") )
( r.kl <- raster.change(r1, r2, s = d, stat = "divergence") )

  opar <- par(no.readonly=TRUE)
  par(mfrow=c(3,2))
    plot(r.kappa, main="Kappa")
    plot(r.ttest[[1]], main="Paired t-test")
    plot(r.ent, main="Delta Entropy")
```

```
    plot(r.cor, main="Rank Correlation")
    plot(r.kl, main="Kullback-Leibler")
    plot(r.ce, main="cross-entropy")
  par(opar)
```

`raster.deviation` *Raster local deviation from the global trend*

Description

Calculates the local deviation from the raster, a specified global statistic or a polynomial trend of the raster.

Usage

```
raster.deviation(
  x,
  type = c("trend", "min", "max", "mean", "median"),
  s = 3,
  degree = 1,
  global = FALSE
)
```

Arguments

`x`	A terra SpatRaster object
`type`	The global statistic to represent the local deviation options are: "trend", "min", "max", "mean", "median"
`s`	Size of matrix (focal window), not used with type="trend"
`degree`	The polynomial degree if type is trend, default is 1st order.
`global`	Use single global value for deviation or cell-level values (FALSE/TRUE). Argument is ignored for type="trend"

Details

The deviation from the trend is derived as [y-hat - y] where; y-hat is the Nth-order polynomial. Whereas the deviation from a global statistic is [y - y-hat] where; y-hat is the local (focal) statistic. The global = TRUE argument allows one to evaluate the local deviation from the global statistic [stat(x) - y-hat] where; stat(x) is the global value of the specified statistic and y-hat is the specified focal statistic.

Value

A SpatRaster class object representing local deviation from the raster or the specified global statistic

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

References

Magee, Lonnie (1998). Nonlocal Behavior in Polynomial Regressions. The American Statistician. American Statistical Association. 52(1):20-22

Fan, J. (1996). Local Polynomial Modelling and Its Applications: From linear regression to nonlinear regression. Monographs on Statistics and Applied Probability. Chapman and Hall/CRC. ISBN 0-412-98321-4

Examples

```
library(terra)
elev <- rast(system.file("extdata/elev.tif", package="spatialEco"))

# local deviation from first-order trend, global mean and raw value
r.dev.trend <- raster.deviation(elev, type="trend", degree=1)
r.dev.mean <- raster.deviation(elev, type="mean", s=5)
r.gdev.mean <- raster.deviation(elev, type="mean", s=5, global=TRUE)

opar <- par(no.readonly=TRUE)
par(mfrow=c(2,2))
  plot(elev, main="original")
  plot(r.dev.trend, main="dev from trend")
  plot(r.dev.mean, main="dev of mean from raw values")
  plot(r.gdev.mean, main="local dev from global mean")
par(opar)
```

`raster.downscale` *Raster Downscale*

Description

Downscales a raster to a higher resolution raster using a robust regression

Usage

```
raster.downscale(
  x,
  y,
  scatter = FALSE,
  full.res = FALSE,
  residuals = FALSE,
  se = FALSE,
  p = 0.95,
  uncertainty = c("none", "prediction", "confidence")
)
```

Arguments

`x`	A terra SpatRaster object representing independent variable(s)
`y`	A terra SpatRaster object representing dependent variable
`scatter`	(FALSE/TRUE) Optional scatter plot
`full.res`	(FALSE/TRUE) Use full resolution of x (see notes)
`residuals`	(FALSE/TRUE) Output raster residual error raster, at same resolution as y
`se`	(FALSE/TRUE) Output standard error raster, using prediction or confidence interval
`p`	The confidence/prediction interval (default is 95%)
`uncertainty`	Output uncertainty raster(s) of confidence or prediction interval, at same resolution as y. Options are c("none", "prediction", "confidence")

Details

This function uses a robust regression, fit using an M-estimation with Tukey's biweight initialized by a specific S-estimator, to downscale a raster based on higher-resolution or more detailed raster data specified as covariate(s). You can optionally output residual error, standard error and/or uncertainty rasters. However, please note that when choosing the type of uncertainty, using a confidence interval (uncertainty around the mean predictions) when you should be using the prediction interval (uncertainty around a single values) will greatly underestimate the uncertainty in a given predicted value (Bruce & Bruce 2017). The full.res = TRUE option uses the x data to sample y rather than y to sample x. THis makes the problem much more computationally and memory extensive and should be used with caution. There is also the question of pseudo-replication (sample redundancy) in the dependent variable. Statistically speaking one would expect to capture the sample variation of x by sampling at the frequency of y thus supporting the downscaling estimate. Note that if uncertainty is not defined the prediction interval for standard error defaults to "confidence" else is the same output as uncertainty (eg., prediction or confidence).

Value

A list object containing:

- downscale - downscaled terra SpatRaster object
- model - MASS rlm model object
- MSE - Mean Square Error
- AIC - Akaike information criterion
- parm.ci - Parameter confidence intervals
- residuals - If residuals = TRUE, a SpatRaster of the residual error
- uncertainty - If pred.int = TRUE, SpatRaster's of the lower/upper prediction intervals
- std.error - If se = TRUE, SpatRaster's of the standard error

Author(s)

Jeffrey S. Evans jeffrey_evans@tnc.org

References

Bruce, P., & A. Bruce. (2017). Practical Statistics for Data Scientists. O'Reilly Media.

Examples

```
if (require(geodata, quietly = TRUE)) {
library(terra)
library(geodata)

# Download example data (requires geodata package)
elev <- elevation_30s(country="SWZ", path=tempdir())
slp <- terrain(elev, v="slope")
tmax <- worldclim_country(country="SWZ", var="tmax", path=tempdir())
  tmax <- crop(tmax[[1]], ext(elev))

# Downscale temperature
x=c(elev,slp)
  names(x) <- c("elev","slope")
y=tmax
  names(y) <- c("tmax")

tmax.ds <- raster.downscale(x, y, scatter=TRUE, residuals = TRUE,
                            uncertainty = "prediction", se = TRUE)

  # plot prediction and parameters
  opar <- par(no.readonly=TRUE)
    par(mfrow=c(2,2))
      plot(tmax, main="Temp max")
      plot(x[[1]], main="elevation")
      plot(x[[2]], main="slope")
      plot(tmax.ds$downscale, main="Downscaled Temp max")
  par(opar)

  # Plot residual error and raw prediction +/- intervals
  opar <- par(no.readonly=TRUE)
    par(mfrow=c(2,2))
      plot(tmax.ds$std.error, main="Standard Error")
      plot(tmax.ds$residuals, main="residuals")
      plot(tmax.ds$uncertainty[[1]],
       main="lower prediction interval")
      plot(tmax.ds$uncertainty[[2]],
       main="upper prediction interval")
  par(opar)

  # plot prediction uncertainty
  opar <- par(no.readonly=TRUE)
    par(mfrow=c(2,1))
      plot(tmax.ds$downscale - tmax.ds$uncertainty[[1]],
       main="lower prediction interval")
      plot(tmax.ds$downscale - tmax.ds$uncertainty[[2]],
       main="upper prediction interval")
```

```
  par(opar)

} else {
  cat("Please install geodata package to run example", "\n")
}
```

`raster.entropy` *Raster Entropy*

Description

Calculates entropy on integer raster (i.e., 8 bit 0-255)

Usage

```
raster.entropy(x, d = 5, categorical = FALSE, global = FALSE, ...)
```

Arguments

`x`	A terra SpatRaster object (requires integer raster)
`d`	Size of matrix (window)
`categorical`	Is the data categorical or continuous (FALSE/TRUE)
`global`	Should the model use a global or local n to calculate entropy (FALSE/TRUE)
`...`	Optional arguments passed terra focal function

Details

Entropy calculated as: H = -sum(Pi*ln(Pi)) where; Pi, Proportion of one value to total values Pi=n(p)/m and m, Number of unique values. Expected range: 0 to log(m) H=0 if window contains the same value in all cells. H increases with the number of different values in the window. The ellipsis arguments can be used to write to disk using the filename argument.

Value

terra SpatRaster class object

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

References

Fuchs M., R. Hoffmann, F. Schwonke (2008) Change Detection with GRASS GIS - Comparison of images taken by different sensor.

Examples

```
library(terra)
  r <- rast(ncols=100, nrows=100)
    r[] <- round(runif(ncell(r), 1,8), digits=0)

rEnt <- raster.entropy(r, d=5, categorical = TRUE, global = TRUE)
  opar <- par(no.readonly=TRUE)
    par(mfcol=c(2,1))
      plot(r)
        plot(rEnt)
  par(opar)

# Maximum entropy is reached when all values are different, same as log(m)
#   for example; log( length( unique(x) ) )
```

`raster.gaussian.smooth` *Gaussian smoothing of raster*

Description

Applies a Gaussian smoothing kernel to smooth raster.

Usage

```
raster.gaussian.smooth(
  x,
  s = 2,
  n = 5,
  scale = FALSE,
  type = c("mean", "median", "sd", "convolution"),
  ...
)
```

Arguments

`x`	A terra SpatRaster raster object
`s`	Standard deviation (sigma) of kernel (default is 2)
`n`	Size of the focal matrix, single value (default is 5 for 5x5 window)
`scale`	(FALSE/TRUE) Scale sigma to the resolution of the raster
`type`	The statistic to use in the smoothing operator; "mean", "median", "sd", "convolution"
`...`	Additional arguments passed to terra::focal

Details

This applies a Gaussian Kernel smoother. The convolution option performs a Gaussian decomposition whereas the other options use the kernel as weights for the given statistic.

Value

A terra SpatRaster class object of the local distributional moment

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
library(terra)
elev <- rast(system.file("extdata/elev.tif", package="spatialEco"))

# Calculate Gaussian smoothing with sigma = 2 and 7x7 window
g1 <- raster.gaussian.smooth(elev, s=2, n=7)
    plot(c(elev,g1))
```

`raster.invert` *Invert raster*

Description

Inverts (flip) the values of a raster

Usage

```
raster.invert(x)
```

Arguments

`x`	A terra SpatRaster object

Details

Inverts raster values using the formula: (((x - max(x)) * -1) + min(x)

Value

A terra SpatRaster object with inverted (flipped) raster values

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
library(terra)
r <- rast(nrows=500, ncols=500, xmin=571823, xmax=616763,
           ymin=4423540, ymax=4453690)
 crs(r) <- "epsg:9001"
r[] <- runif(ncell(r), 1000, 2500)
r <- focal(r, focalMat(r, 150, "Gauss") )

r.inv <- raster.invert(r)

opar <- par(no.readonly=TRUE)
    par(mfrow=c(1,2))
      plot(r, main="original raster")
      plot(r.inv, main="inverted raster")
par(opar)
```

`raster.kendall` *Kendall tau trend with continuity correction for raster time-series*

Description

Calculates a nonparametric statistic for a monotonic trend based on the Kendall tau statistic and the Theil-Sen slope modification

Usage

```
raster.kendall(
  x,
  intercept = TRUE,
  p.value = TRUE,
  confidence = TRUE,
  tau = TRUE,
  min.obs = 6,
  method = c("zhang", "yuepilon", "none"),
  ...
)
```

Arguments

`x`	A multiband terra SpatRaster object with at least 5 layers
`intercept`	(FALSE/TRUE) return a raster with the pixel wise intercept values
`p.value`	(FALSE/TRUE) return a raster with the pixel wise p.values
`confidence`	(FALSE/TRUE) return a raster with the pixel wise 95 pct confidence levels
`tau`	(FALSE/TRUE) return a raster with the pixel wise tau correlation values
`min.obs`	The threshold of minimum number of observations (default 6)
`method`	Kendall method to use c("zhang", "yuepilon","none"), see kendall function
`...`	Additional arguments passed to the terra app function

Details

This function implements Kendall's nonparametric test for a monotonic trend using the Theil-Sen (Theil 1950; Sen 1968; Siegel 1982) method to estimate the slope and related confidence intervals.

Value

Depending on arguments, a raster layer or rasterBrick object containing:

- raster layer 1 - slope for trend, always returned
- raster layer 2 - Kendall's tau two-sided test, reject null at 0, if tau TRUE
- raster layer 3 - intercept for trend if intercept TRUE
- raster layer 4 - p value for trend fit if p.value TRUE
- raster layer 5 - lower confidence level at 95 pct, if confidence TRUE
- raster layer 6 - upper confidence level at 95 pct, if confidence TRUE

Author(s)

Jeffrey S. Evans jeffrey_evans@tnc.org

References

Theil, H. (1950) A rank invariant method for linear and polynomial regression analysis. Nederl. Akad. Wetensch. Proc. Ser. A 53:386-392 (Part I), 53:521-525 (Part II), 53:1397-1412 (Part III).

Sen, P.K. (1968) Estimates of Regression Coefficient Based on Kendall's tau. Journal of the American Statistical Association. 63(324):1379-1389.

Siegel, A.F. (1982) Robust Regression Using Repeated Medians. Biometrika, 69(1):242-244

See Also

`zyp.trend.vector` for model details

`app` for available ... arguments

Examples

```
library(terra)

# note; nonsense example with n=9
r <- c(rast(system.file("ex/logo.tif", package="terra")),
       rast(system.file("ex/logo.tif", package="terra")),
       rast(system.file("ex/logo.tif", package="terra")))

# Calculate trend slope with p-value and confidence level(s)
# ("slope","intercept", "p.value","z.value", "LCI","UCI","tau")
  k <- raster.kendall(r, method="none")
  plot(k)
```

`raster.mds` *Raster multidimensional scaling (MDS)*

Description

Multidimensional scaling of raster values within an N x N focal window

Usage

```
raster.mds(r, s = 5, window.median = FALSE, ...)
```

Arguments

`r`	A terra SpatRaster class object
`s`	Window size (may be a vector of 1 or 2) of n x n dimension.
`window.median`	(FALSE/TRUE) Return the median of the MDS matrix values.
`...`	Additional arguments passed to terra::focal

Details

An MDS focal function. If only one value provided for s, then a square matrix (window) will be used. If window.median = FALSE then the center value of the matrix is returned and not the median of the matrix

Value

A terra SpatRaster class object

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

References

Quinn, G.P., & M.J. Keough (2002) Experimental design and data analysis for biologists. Cambridge University Press. Ch. 18. Multidimensional scaling and cluster analysis.

Examples

```
 library(terra)
 r <- rast(system.file("ex/elev.tif", package="terra"))
   r <- r[[1]] / max(global(r, "max", na.rm=TRUE)[,1])

 diss <- raster.mds(r)
 diss.med <- raster.mds(r, window.median = TRUE)

opar <- par(no.readonly=TRUE)
```

```
  par(mfrow=c(2,2))
  plot(r)
    title("Elevation")
  plot( focal(r, w = matrix(1, nrow=5, ncol=5), fun = var) )
    title("Variance")
    plot(diss)
      title("MDS")
    plot(diss.med)
      title("Median MDS")
par(opar)
```

`raster.modified.ttest` *Dutilleul moving window bivariate raster correlation*

Description

A bivarate raster correlation using Dutilleul's modified t-test

This function provides a bivariate moving window correlation using the modified t-test to account for spatial autocorrelation. Point based subsampling is provided for computation tractability. The hexagon sampling is recommended as it it good at capturing spatial process that includes nonstationarity and anistropy.

Usage

```
raster.modified.ttest(
  x,
  y,
  d = "auto",
  sample = c("none", "random", "hexagonal", "regular"),
  p = 0.1,
  size = NULL
)
```

Arguments

`x`	A terra SpatRaster class object
`y`	A terra SpatRaster class object, same dimensions as x
`d`	Distance for finding neighbors
`sample`	Apply sub-sampling options; c("none", "random", "hexagonal", "regular")
`p`	If sample != "none", what proportion of population should be sampled
`size`	Fixed sample size (default NULL)

Value

A terra SpatRaster or sf POINT class object with the following attributes:

- corr - Correlation
- Fstat - The F-statistic calculated as degrees of freedom unscaled F-statistic
- p.value - p-value for the test
- moran.x - Moran's-I for x
- moran.y - Moran's-I for y

Author(s)

Jeffrey S. Evans jeffrey_evans@tnc.org

References

Clifford, P., S. Richardson, D. Hemon (1989), Assessing the significance of the correlationbetween two spatial processes. Biometrics 45:123-134.

Dutilleul, P. (1993), Modifying the t test for assessing the correlation between two spatial processes. Biometrics 49:305-314.

See Also

`modified.ttest` for test details

Examples

```
 p = c("sf", "sp", "terra", "gstat")
 if(any(!unlist(lapply(p, requireNamespace, quietly=TRUE)))) {
   m = which(!unlist(lapply(p, requireNamespace, quietly=TRUE)))
   message("Can't run examples, please install ", paste(p[m], collapse = " "))
 } else {
   invisible(lapply(p, require, character.only=TRUE))

data(meuse, package = "sp")
meuse <- st_as_sf(meuse, coords = c("x", "y"), crs = 28992,
                  agr = "constant")
data(meuse.grid, package = "sp")
meuse.grid <- st_as_sf(meuse.grid, coords = c("x", "y"), crs = 28992,
                  agr = "constant")

ref <- rast(ext(meuse.grid), resolution = 40)
  crs(ref) <- crs(meuse)
e <- ext(179407.8, 181087.9, 331134.4, 332332.1)

# GRID-1 log(copper):
v1 <- variogram(log(copper) ~ 1, meuse)
  x1 <- fit.variogram(v1, vgm(1, "Sph", 800, 1))
  G1 <- krige(zinc ~ 1, meuse, meuse.grid, x1, nmax = 30)
G1 <- crop(rasterize(G1, ref, "var1.pred"),e)
```

```
names(G1) <- "copper"

 # GRID-2 log(elev):
v2 <- variogram(log(elev) ~ 1, meuse)
  x2 <- fit.variogram(v2, vgm(1, "Sph", 800, 1))
  G2 <- krige(zinc ~ 1, meuse, meuse.grid, x2, nmax = 30)
G2 <- crop(rasterize(G2, ref, "var1.pred"),e)
names(G2) <- "elev"

# Raster corrected correlation
acor <- raster.modified.ttest(G1, G2)
  plot(acor)

# Sample-based corrected correlation
( cor.hex <- raster.modified.ttest(G1, G2, sample = "hexagonal") )
  plot(cor.hex["corr"], pch=20)
}
```

`raster.moments` *Raster moments*

Description

Calculates focal statistical moments of a raster

Usage

```
raster.moments(x, type = "mean", s = 3, p = 0.75, ...)
```

Arguments

`x`	A terra SpatRaster object
`type`	The global statistic to represent the local deviation options are: "min", "min", "mean", "median", "var, "sd", "mad", "kurt", "skew", "quantile"
`s`	Size of matrix (focal window), can be single value or two values defining the [x,y] dimensions of the focal matrix
`p`	if type="quantile", the returned percentile.
`...`	Additional arguments passed to terra::focal

Details

This is a simple wrapper for the terra focal function, returning local statistical moments

Value

A terra SpatRaster object representing the local distributional moment

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
library(terra)
r <- rast(nrows=500, ncols=500, xmin=571823, xmax=616763,
            ymin=4423540, ymax=4453690)
  crs(r) <- "epsg:9001"
r[] <- runif(ncell(r), 1000, 2500)

# Calculate 10th percentile for 3x3 window
r.p10 <- raster.moments(r, type="quantile", p=0.10)
```

`raster.transformation` *Statistical transformation for rasters*

Description

Transforms raster to a specified statistical transformation

Usage

```
raster.transformation(x, trans = "norm", smin = 0, smax = 255)
```

Arguments

`x`	A terra SpatRaster class object
`trans`	Transformation method: "norm", "rstd", "std", "stretch", "nl", "slog", "sr" (please see notes)
`smin`	Minimum value for stretch
`smax`	Maximum value for stretch

Details

Transformation option details:

- norm - (Normalization_ (0-1): if min(x) < 0 (x - min(x)) / (max(x) - min(x))
- rstd - (Row standardize) (0-1): if min(x) >= 0 x / max(x) This normalizes data
- `with negative distributions`
- std - (Standardize) (x - mean(x)) / sdv(x)
- stretch - (Stretch) ((x - min(x)) * max.stretch / (max(x) - min(x)) + min.stretch) This will stretch values to the specified minimum and maximum values (eg., 0-255 for 8-bit)

- nl - (Natural logarithms) if min(x) > 0 log(x)
- slog - (Signed log 10) (for skewed data): if min(x) >= 0 ifelse(abs(x) <= 1, 0, sign(x)*log10(abs(x)))
- sr - (Square-root) if min(x) >= 0 sqrt(x)

Value

A terra SpatRaster class object of specified transformation

Author(s)

Jeffrey S. Evans jeffrey_evans@tnc.org

Examples

```
library(terra)
r <- rast(nrows=500, ncols=500, xmin=571823, xmax=616763,
            ymin=4423540, ymax=4453690)
  crs(r) <- "epsg:9001"
r[] <- runif(ncell(r), 1000, 2500)

 # Positive values so, can apply any transformation
 for( i in c("norm", "rstd", "std", "stretch", "nl", "slog", "sr")) {
   print( raster.transformation(r, trans = i) )
 }

 # Negative values so, can't transform using "nl", "slog" or "sr"
 r[] <- runif(ncell(r), -1, 1)
   for( i in c("norm", "rstd", "std", "stretch", "nl", "slog", "sr")) {
  try( print( raster.transformation(r, trans = i) ) )
   }
```

---

raster.vol *Raster Percent Volume*

Description

Calculates a percent volume on a raster or based on a systematic sample

Usage

```
raster.vol(
  x,
  p = 0.75,
  sample = FALSE,
  spct = 0.05,
  type = c("random", "regular")
)
```

Arguments

`x`	A terra SpatRaster class object
`p`	percent raster-value volume
`sample`	(FALSE/TRUE) base volume on systematic point sample
`spct`	sample percent, if sample (TRUE)
`type`	If sample=TRUE type of sample, options are "random" or "regular"

Value

if sample (FALSE) binary raster object with 1 representing designated percent volume else, if sample (TRUE) n sf POINT object with points that represent the percent volume of the sub-sample

Note

Since this model needs to operate on all of the raster values, it is not memory safe

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
library(terra)
  r <- rast(ncols=100, nrows=100)
    r[] <- runif(ncell(r), 0, 1)
    r <- focal(r, w=focalMat(r, 6, ”Gauss”))
  #r[sample(1:ncell(r)),10] <- NA

  # full raster percent volume
  p30 <- raster.vol(r, p=0.30)
  p50 <- raster.vol(r, p=0.50)
  p80 <- raster.vol(r, p=0.80)

opar <- par(no.readonly=TRUE)
    par(mfrow=c(2,2))
    plot(r, col=cm.colors(10), main=”original raster”)
    plot(p30, breaks=c(0,0.1,1), col=c(”cyan”,”red”), legend=FALSE,
      main=”30% volume”)
    plot(p50, breaks=c(0,0.1,1), col=c(”cyan”,”red”), legend=FALSE,
      main=”50% volume”)
    plot(p80, breaks=c(0,0.1,1), col=c(”cyan”,”red”), legend=FALSE,
      main=”80% volume”)
par(opar)
```

`raster.Zscore` *Modified z-score for a raster*

Description

Calculates the modified z-score for raster values

Usage

```
raster.Zscore(x, p.value = FALSE, file.name = NULL, ...)
```

Arguments

`x`	A raster class object
`p.value`	Return p-value rather than z-score raster (FALSE/TRUE)
`file.name`	Name of raster written to disk
`...`	Additional arguments passed to writeRaster

Value

raster class object or raster written to disk

Note

Since this functions needs to operate on all of the raster values, it is not memory safe

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```

library(terra)
r <- rast(nrows=500, ncols=500)
  r[] <- runif(ncell(r), 0, 1)

# Modified z-score
( z <- raster.Zscore(r) )

# P-value
( p <- raster.Zscore(r, p.value = TRUE) )
```

`rasterCorrelation` *Raster correlation*

Description

Performs a moving window correlation between two rasters

Usage

```
rasterCorrelation(x, y, s = 3, type = "pearson")
```

Arguments

`x`	A terra SpatRaster class object for x
`y`	A terra SpatRasterclass object for y
`s`	Scale of window. Can be a single value, two values for uneven window or a custom matrix. Must be odd number (eg., s=3, for 3x3 window or s=c(3,5) for 3 x 5 window)
`type`	Type of output, options are: "pearson", "spearman", "covariance"

Value

A terra SpatRaster class object

Note

The NA behavior is set to na.rm = TRUE to make default outputs consistent between the terra and raster packages.

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
library(terra)

r <- rast(system.file("ex/logo.tif", package="terra"))
  x <- r[[1]]
  y <- r[[3]]

 r.cor <- rasterCorrelation(x, y, s = 5, type = "spearman")
   plot(r.cor)
```

`rasterDistance` *Raster Distance*

Description

Calculates the Euclidean distance of a defined raster class and all the other cells in a taster

Usage

```
rasterDistance(x, y, scale = FALSE)
```

Arguments

`x`	A terra SpatRast or sf class object
`y`	Value(s) in x to to calculate distance to
`scale`	(FALSE/TRUE) Perform a row standardization on results

Details

This replicates the terra distance function but uses the Arya & Mount Approximate Near Neighbor (ANN) C++ library for calculating distances. Where this results in a notable increase in performance it is not memory safe, needing to read in the entire raster and does not use the GeographicLib (Karney, 2013) spheroid distance method for geographic data.

Value

A terra SpatRast raster representing distances

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

References

Arya S., Mount D. M., Netanyahu N. S., Silverman R. and Wu A. Y (1998), An optimal algorithm for approximate nearest neighbor searching, Journal of the ACM, 45, 891-923.

See Also

`distance`, `distance`

Examples

```
library(sf)
library(terra)

# read, project and subset 10 polygons
nc <- suppressWarnings(st_cast(st_read(system.file("shape/nc.shp",
         package="sf")), "POLYGON"))
  nc <- st_transform(nc, st_crs("ESRI:102008"))
    nc.sub <- nc[sample(1:nrow(nc),10),]

# create 1000m reference raster, rasterize subset polygons
ref <- rast(ext(nc), resolution=1000)
  rnc <- mask(rasterize(vect(nc.sub), field="CNTY_ID",
              ref, background=9999), vect(nc))
    crs(rnc) <- "ESRI:102008"

# Calculate distance to class 1 in rnc raster, plot results
ids <- nc.sub$CNTY_ID
rd <- rasterDistance(rnc, y=ids)
  plot(rd)
    plot( st_geometry(nc.sub), add=TRUE)
```

`remove.holes` *Remove or return polygon holes*

Description

Removes or returns all holes (null geometry) in sf polygon class objects

Usage

```
remove.holes(x, only.holes = FALSE)
```

Arguments

`x`	sf POLYGON or MULTIPOLYGON object
`only.holes`	Delete holes (FALSE) or returns only holes (FALSE)

Details

A hole is considered a polygon within a polygon (island) representing null geometry. If you want to return only holes, no longer NULL, use keep = TRUE. To delete holes use default only.holes = FALSE. Single part features will be returned regardless of input.

Value

sf POLYGON object

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
library(sf)

p <- sf::st_as_sf(sf::st_sfc(
  sf::st_polygon(list(
  cbind(c(2,4,4,1,2),c(2,3,5,4,2)),
  cbind(c(2.33, 2.05, 3.25, 3.25, 2.33),
      c(3.00, 3.56, 3.95, 3.46, 3.00)))),
  sf::st_polygon(list(
    cbind(c(5,4,2,5),c(2,3,2,2)))),
  sf::st_polygon(list(
    cbind(c(4,4,5,10,4),c(5,3,2,5,5)),
    cbind(c(5,6,6,5,5),c(4,4,3,3,4))
))))
  p$ID <- 1:3

rh <- remove.holes(p)
kh <- remove.holes(p, only.holes=TRUE)

opar <- par(no.readonly=TRUE)
   par(mfrow=c(2,2))
     plot(st_geometry(p), main="Original with holes")
     plot(st_geometry(rh), main="holes removed only.holes=FALSE")
 plot(st_geometry(kh), main="return holes only.holes=TRUE")
par(opar)
```

`remove_duplicates` *Remove duplicate geometries*

Description

Removes duplicate geometries in a single-part feature class

Usage

```
remove_duplicates(x, threshold = 0.00001)
```

Arguments

`x`	An sf POINT, POLYGON or LINESTRING object
`threshold`	A distance threshold indicating fuzzy duplication, default i 0.00001

Details

This function removes duplicate geometries based on order and not "non null" attribution or other factors, the first feature gets to stay. If one needs to know which points were removed sf::st_difference can be used between original data and results of the function.

Value

sf object, of same feature class as x, with duplicate geometries removed

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
library(sf)

# data with 10 duplicate obs
s <- data.frame(x = runif(100), y = runif(100))
  s <- data.frame(rbind(s, s[sample(1:nrow(s), 10),]) )
    s <- st_as_sf(s, coords = c("x", "y"))
      s$ID <- 1:nrow(s)

nrow(s)
nrow( srmd <- remove_duplicates(s) )
```

`rm.ext` *Remove extension*

Description

Removes file extension (and path) from string

Usage

```
rm.ext(x)
```

Arguments

`x`	A character vector representing a file with extension

Value

The file name with extension and file path stripped off

Examples

```
rm.ext("C:/path/file.txt")
```

`rotate.polygon` *Rotate polygon*

Description

rotates polygon by specified angle

Usage

```
rotate.polygon(
  p,
  angle = 45,
  sp = FALSE,
  anchor = c("center", "lower.left", "upper.right")
)
```

Arguments

`p`	A polygon object of sf or sp class
`angle`	Rotation angle in degrees
`sp`	(FALSE | TRUE) Output sp class object
`anchor`	Location to rotate polygon on options are "center", "lower.left" and "upper.right"

Details

The anchor is the location that the rotation is anchored to. The center is the centroid where the lower.left and upper.right are based on the min or max of the coordinates respectively.

Value

an sp or sf polygon object with rotated polygon

Examples

```
library(sf)

data(meuse, package = "sp")
  meuse <- st_as_sf(meuse, coords = c("x", "y"),
                    crs = 28992, agr = "constant")

 e <- st_convex_hull(st_union(meuse))
   e30 <- rotate.polygon(e, angle=30)

 plot(e, main="rotated 30 degrees")
   plot(e30, add=TRUE)
```

`sa.trans` *Trigonometric transformation of a slope and aspect interaction*

Description

The Trigonometric Stage (1978) [slope * cos(aspect)] or [slope * sin(aspect)]

Usage

```
sa.trans(
  slope,
  aspect,
  type = "cos",
  slp.units = "degrees",
  asp.units = "degrees"
)
```

Arguments

`slope`	slope values in degrees, radians or percent
`aspect`	aspect values in degrees or radians
`type`	Type of transformation, options are: "cos", "sin"
`slp.units`	Units of slope values, options are: "degrees", "radians" or "percent"
`asp.units`	Units of aspect values, options are: "degrees" or "radians"

Details

An a priori assumption of a maximum in the NW quadrant (45 azimuth) and a minimum in the SW quadrant can be replaced by an empirically determined location of the optimum without repeated calculations of the regression fit. In addition it is argued that expressions for the effects of aspect should always be considered as terms involving an interaction with slope (Stage, 1976)

For slopes from 0 bounded from -1 to 1. Greater than 100 out of the -1 to 1 range.

An alternative for slopes with values approaching infinity is to take the square root of slope/100 to reduce the range of values.By default this model test all values greater than 100 to 101

Value

A vector of the modeled value

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

References

Stage, A. R. 1976. An Expression of the Effects of Aspect, Slope, and Habitat Type on Tree Growth. Forest Science 22(3):457-460.

Examples

```
library(terra)
sa.trans(slope = 48.146, aspect = 360.000)

# Example of creating slope*cos(aspect) raster
elev <- rast(system.file("extdata/elev.tif", package="spatialEco"))
sa <- terra::terrain(elev, v=c("slope", "aspect"), unit="degrees")
scosa <- terra::lapp(c(sa[[1]], sa[[2]]), fun = sa.trans)
```

`sample.annulus` *Sample annulus*

Description

Creates sample points based on annulus with defined inner and outer radius

Usage

```
sample.annulus(x, r1, r2, size = 10, ...)
```

Arguments

`x`	An sf POINT class object
`r1`	Numeric value defining inner radius of annulus (in projection units)
`r2`	Numeric value defining outer radius of annulus (in projection units)
`size`	Number of samples
`...`	Additional arguments passed to sf::st_sample

Details

Function can be used for distance based sampling which is a sampling method that can be used to capture spatially lagged variation.

Value

sf POINTS object

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
 library(sf)
 if(require(sp, quietly = TRUE)) {
   data(meuse, package = "sp")
   meuse <- st_as_sf(meuse, coords = c("x", "y"), crs = 28992,
                     agr = "constant")

 xy <- meuse[2,]
 rs100 <- sample.annulus(xy, r1=50, r2=100, size = 50)
 rs200 <- sample.annulus(xy, r1=100, r2=200, size = 50)

 plot(st_geometry(rs200), pch=20, col="red")
   plot(st_geometry(rs100), pch=20, col="blue", add=TRUE)
   plot(st_geometry(xy), pch=20, cex=2, col="black", add=TRUE)
 legend("topright", legend=c("50-100m", "100-200m", "source"),
        pch=c(20,20,20), col=c("blue","red","black"))


# Run on multiple points
rs100 <- sample.annulus(meuse[1:3,], r1=50, r2=100,
                        size = 50)
rs200 <- sample.annulus(meuse[1:3,], r1=50, r2=200,
                        size = 50)
plot(st_geometry(rs200), pch=20, col="red")
  plot(st_geometry(rs100), pch=20, col="blue", add=TRUE)
    plot(st_geometry(meuse[1:3,]), pch=20, cex=2, col="black", add=TRUE)
 legend("topright", legend=c("50-100m", "100-200m", "source"),
        pch=c(20,20,20), col=c("blue","red","black"))

} else {
  cat("Please install sp package to run example", "\n")
}
```

`sampleTransect` *Sample transect*

Description

Creates random transects from points and generates sample points along each transect

Usage

```
sampleTransect(
  x,
  min.dist,
  max.dist,
  distance = NULL,
  azimuth = NULL,
```

```
  id = NULL,
  ...
)
```

Arguments

`x`	A sf point object
`min.dist`	Minimum length of transect(s)
`max.dist`	Maximum length of transect(s)
`distance`	A vector of distances, same length as x, used to define transect distances (length)
`azimuth`	A vector of azimuths, same length as x, used to define transect direction
`id`	A unique identification column in x
`...`	Additional arguments passed to st_sample

Details

Function create lines and samples using random or defined direction and length transects and then creates a point sample along each transect. The characteristic of the sample points are defined by arguments passed to the sf::st_sample function. The distance and azimuth arguments allow for specifying the exact length and direction for each points transect.

Value

A list object contaning sf LINES and POINTS objects representing random transects and sample points along each transect. The "ID" column links the resulting data.

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
if(require(sp, quietly = TRUE)) {
library(sf)
  data(meuse, package = "sp")
  meuse <- st_as_sf(meuse, coords = c("x", "y"), crs = 28992,
                    agr = "constant")
meuse <- meuse[sample(1:nrow(meuse),10),]

transects <- sampleTransect(meuse, min.dist=200, max.dist=500,
                            type="regular", size=20)
   plot(st_geometry(transects$transects))
     plot(st_geometry(meuse), pch=19, cex=2, add=TRUE)
       plot(st_geometry(transects$samples),
        col="red", pch=19, add=TRUE)

} else {
  cat("Please install sp package to run example", "\n")
}
```

`sar` *Surface Area Ratio*

Description

Calculates the Berry (2002) Surface Area Ratio based on slope

Usage

```
sar(x, s = NULL, scale = TRUE)
```

Arguments

`x`	A terra SpatRaster object
`s`	cell resolution (default is NULL and not needed if projection is in planar units)
`scale`	(TRUE/FALSE) Scale (row standardize) results

Details

SAR is calculated as: resolution^2 * cos((degrees(slope) * (pi / 180)))

Value

A terra SpatRaster class object of the Surface Area Ratio

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

References

Berry, J.K. (2002). Use surface area for realistic calculations. Geoworld 15(9):20-1.

Examples

```
library(terra)
elev <- rast(system.file("extdata/elev.tif", package="spatialEco"))
( surface.ratio <- sar(elev) )
  plot(surface.ratio)
```

`separability` *separability*

Description

Calculates variety of two-class sample separability metrics

Usage

```
separability(
  x,
  y,
  plot = FALSE,
  cols = c("red", "blue"),
  clabs = c("Class1", "Class2"),
  ...
)
```

Arguments

`x`	X vector
`y`	Y vector
`plot`	plot separability (TRUE/FALSE)
`cols`	colors for plot (must be equal to number of classes)
`clabs`	labels for two classes
`...`	additional arguments passes to plot

Details

Available statistics:

- M-Statistic (Kaufman & Remer 1994) - This is a measure of the difference of the distributional peaks. A large M-statistic indicates good separation between the two classes as within-class variance is minimized and between-class variance maximized (M <1 poor, M >1 good).
- Bhattacharyya distance (Bhattacharyya 1943; Harold 2003) - Measures the similarity of two discrete or continuous probability distributions.
- Jeffries-Matusita distance (Bruzzone et al., 2005; Swain et al., 1971) - The J-M distance is a function of separability that directly relates to the probability of how good a resultant classification will be. The J-M distance is asymptotic to v2, where values of v2 suggest complete separability
- Divergence and transformed Divergence (Du et al., 2004) - Maximum likelihood approach. Transformed divergence gives an exponentially decreasing weight to increasing distances between the classes.

Value

A data.frame with the following separability metrics:

- B - Bhattacharryya distance statistic
- JM - Jeffries-Matusita distance statistic
- M - M-Statistic
- D - Divergence index
- TD - Transformed Divergence index

Author(s)

Jeffrey S. Evans jeffrey_evans@tnc.org

References

Anderson, M. J., & Clements, A. (2000) Resolving environmental disputes: a statistical method for choosing among competing cluster models. Ecological Applications 10(5):1341-1355

Bhattacharyya, A. (1943) On a measure of divergence between two statistical populations defined by their probability distributions'. Bulletin of the Calcutta Mathematical Society 35:99-109

Bruzzone, L., F. Roli, S.B. Serpico (1995) An extension to multiclass cases of the Jefferys-Matusita distance. IEEE Transactions on Pattern Analysis and Machine Intelligence 33:1318-1321

Du, H., C.I. Chang, H. Ren, F.M. D'Amico, J. O. Jensen, J., (2004) New Hyperspectral Discrimination Measure for Spectral Characterization. Optical Engineering 43(8):1777-1786.

Kailath, T., (1967) The Divergence and Bhattacharyya measures in signal selection. IEEE Transactions on Communication Theory 15:52-60

Kaufman Y., and L. Remer (1994) Detection of forests using mid-IR reflectance: An application for aerosol studies. IEEE T. Geosci.Remote. 32(3):672-683.

Examples

```
norm1 <- dnorm(seq(-20,20,length=5000),mean=0,sd=1)
norm2 <- dnorm(seq(-20,20,length=5000),mean=0.2,sd=2)
  separability(norm1, norm2)

s1 <- c(1362,1411,1457,1735,1621,1621,1791,1863,1863,1838)
s2 <- c(1362,1411,1457,10030,1621,1621,1791,1863,1863,1838)
  separability(s1, s2, plot=TRUE)
```

`sf.kde` *Spatial kernel density estimate*

Description

A weighted or unweighted Gaussian Kernel Density estimate for point spatial data

Usage

```
sf.kde(
  x,
  y = NULL,
  bw = NULL,
  ref = NULL,
  res = NULL,
  standardize = FALSE,
  scale.factor = 10000,
  mask = FALSE
)
```

Arguments

`x`	sf POINT object
`y`	Optional values, associated with x coordinates, to be used as weights
`bw`	Distance bandwidth of Gaussian Kernel, must be units of projection
`ref`	A terra SpatRaster, sf class object or c[xmin,xmax,ymin,ymax] vector to estimate the kde extent
`res`	Resolution of raster when ref not SpatRaster
`standardize`	Standardize results to 0-1 (FALSE/TRUE)
`scale.factor`	Numeric scaling factor for the KDE (defaults to 10000), to account for very small estimate values
`mask`	(TRUE/FALSE) mask resulting raster if ref is provided as a SpatRaster

Details

Please note that ks methods for estimation has been reverted to the Gussian method proposed in Venables & Ripley (2002). There was not enought evendence that the Chacon & Duong (2018) multivariate method(s) for bandwidth selection and kernal estimation were suitable for spatial random fields.

Value

a terra SpatRaster class object containing kernel density estimate

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

References

Duong, T. & Hazelton, M.L. (2005) Cross-validation bandwidth matrices for multivariate kernel density estimation. Scandinavian Journal of Statistics, 32, 485-506.

Wand, M.P. & Jones, M.C. (1994) Multivariate plug-in bandwidth selection. Computational Statistics, 9, 97-116.

Venables, W. N. and Ripley, B. D. (2002) Modern Applied Statistics with S. Fourth edition. Springer.

Examples

```
library(sf)
library(terra)

data(meuse, package = "sp")
meuse <- st_as_sf(meuse, coords = c("x", "y"), crs = 28992,
                  agr = "constant")

# Unweighted KDE (spatial locations only) with 40m resoultion
pt.kde <- sf.kde(x = meuse, bw = 1000, standardize = TRUE, res=40)
  plot(pt.kde, main="Unweighted kde")
    plot(st_geometry(meuse), pch=20, col="red", add=TRUE)

# cadmium weighted KDE usign extent with 40m resoultion and 500m and 1000m bw
cadmium.kde.500 <- sf.kde(x = meuse, y = meuse$cadmium, res=40,
                          bw = 500, standardize = TRUE)
cadmium.kde.1000 <- sf.kde(x = meuse, y = meuse$cadmium, res=40,
                          bw = 1000, standardize = TRUE)
  plot(c(cadmium.kde.500, cadmium.kde.1000))
```

`sf_dissolve` *Dissolve polygons*

Description

Dissolve polygon feature calss

Usage

```
sf_dissolve(x, y = NULL, overlaps = FALSE)
```

Arguments

`x`	An sf POLYGON or MULTIPOLYGON object
`y`	An attribute in x to dissolve by, default is NULL
`overlaps`	(FALSE/TRUE) Dissolve overlapping polygons, negates using attribute

Details

If a dissolve attribute is defined, the result will be a MULTIPOLYGON with the grouping attribute column. If y=NULL all polygons will be dissolved into a single attribute, unless there is spatial discontinuity (eg., gaps) in the data. The intent of overlaps=TRUE is to provide functionality for dissolving overlapping polygons and should only be used in this specialized case.

Value

A dissolved POLYGON or MULTIPOLYGON object

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
library(sf)
nc <- st_read(system.file("shape/nc.shp", package="sf"))
  nc$group <- ifelse(nc$CNTY_ <= 1902, 1,
                ifelse(nc$CNTY_ > 1902 & nc$CNTY_ <= 1982, 2,
     ifelse(nc$CNTY_ > 1982, 3, NA)))

# Dissolve by group attribute
d <- sf_dissolve(nc, "group")
  plot(st_geometry(nc), border="grey")
    plot(st_geometry(d), border="red", col=NA,
         lwd=2, add=TRUE)

# Dissolve all polygons
d <- sf_dissolve(nc)
  plot(st_geometry(nc), border="grey")
    plot(st_geometry(d), border="red", col=NA,
         lwd=2, add=TRUE)

# Dissolve overlapping polygons
sq <- function(pt, sz = 1) st_polygon(list(rbind(c(pt - sz),
     c(pt[1] + sz, pt[2] - sz), c(pt + sz), c(pt[1] - sz, pt[2] + sz),
 c(pt - sz))))
pol <- st_sf(box = 1:6, st_sfc(sq(c(4.2,4.2)), sq(c(0,0)), sq(c(1, -0.8)),
          sq(c(0.5, 1.7)), sq(c(3,3)), sq(c(-3, -3))))
 st_geometry(pol) <- "geometry"

plot(pol)

d <- sf_dissolve(pol, overlaps=TRUE)
```

```
plot(d["diss"])
```

`sg.smooth` *Savitzky-Golay smoothing filter*

Description

Smoothing of time-series data using Savitzky-Golay convolution smoothing

Usage

```
sg.smooth(x, f = 4, l = 51, d = 1, na.rm, ...)
```

Arguments

`x`	A vector to be smoothed
`f`	Filter type (default 4 for quartic, specify 2 for quadratic)
`l`	Convolution filter length, must be odd number (default 51). Defines degree of smoothing
`d`	First derivative (default 1)
`na.rm`	NA behavior
`...`	not used

Value

A vector of the smoothed data equal to length of x. Please note; NA values are retained

Author(s)

Jeffrey S. Evans <jeffrey_evans<at>tnc.org>

References

Savitzky, A., and Golay, M.J.E. (1964). Smoothing and Differentiation of Data by Simplified Least Squares Procedures. Analytical Chemistry. 36(8):1627-39

Examples

```
y <- c(0.112220988, 0.055554941, 0.013333187, 0.055554941, 0.063332640, 0.014444285,
       0.015555384, 0.057777140, 0.059999339, 0.034444068, 0.058888242, 0.136665165,
       0.038888458, 0.096665606,0.141109571, 0.015555384, 0.012222088, 0.012222088,
       0.072221428, 0.052221648, 0.087776810,0.014444285, 0.033332966, 0.012222088,
       0.032221869, 0.059999339, 0.011110989, 0.011110989,0.042221759, 0.029999670,
       0.018888680, 0.098887801, 0.016666483, 0.031110767, 0.061110441,0.022221979,
       0.073332526, 0.012222088, 0.016666483, 0.012222088, 0.122220881, 0.134442955,
       0.094443403, 0.128887475, 0.045555055, 0.152220547, 0.071110331, 0.018888680,
```

```
           0.022221979, 0.029999670, 0.035555165, 0.014444285, 0.049999449, 0.074443623,
           0.068888135, 0.062221535, 0.032221869, 0.095554501, 0.143331751, 0.121109776,
           0.065554835, 0.074443623, 0.043332856, 0.017777583, 0.016666483, 0.036666263,
           0.152220547, 0.032221869, 0.009999890, 0.009999890, 0.021110879, 0.025555275,
           0.099998899, 0.015555384, 0.086665712, 0.008888791, 0.062221535, 0.044443958,
           0.081110224, 0.015555384, 0.089999005, 0.082221314, 0.056666043, 0.013333187,
           0.048888352, 0.075554721, 0.025555275, 0.056666043, 0.146665052, 0.118887581,
           0.125554174, 0.024444176, 0.124443069, 0.012222088, 0.126665279, 0.048888352,
           0.046666153, 0.141109571, 0.015555384, 0.114443190)

plot(y, type="l", lty = 3, main="Savitzky-Golay with l = 51, 25, 10")
  lines(sg.smooth(y),col="red", lwd=2)
  lines(sg.smooth(y, l = 25),col="blue", lwd=2)
  lines(sg.smooth(y, l = 10),col="green", lwd=2)

#### function applied to a multi-band raster
library(terra)
( r <- spatialEco::random.raster(n.layers=20) )

# raster stack example
( r.sg <- app(r, sg.smooth) )
```

`shannons` *Shannon's Diversity (Entropy) Index*

Description

Calculates Shannon's Diversity Index and Shannon's Evenness Index

Usage

```
shannons(x, counts = TRUE, ens = FALSE, margin = "row")
```

Arguments

`x`	data.frame object containing counts or proportions
`counts`	Are data counts (TRUE) or relative proportions (FALSE)
`ens`	Calculate effective number of species (TRUE/FALSE)
`margin`	Calculate diversity for rows or columns. c("row", "col")

Details

The expected for H is 0-3+ where a value of 2 has been suggested as medium-high diversity, for evenness is 0-1 with 0 signifying no evenness and 1, complete evenness.

Value

data.frame with "H" (Shannon's diversity) and "evenness" (Shannon's evenness where H / max(sum(x))) and ESN

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

References

Shannon, C. E. and W. Weaver (1948) A mathematical theory of communication. The Bell System Technical Journal, 27:379-423.

Simpson, E. H. (1949) Measurement of diversity. Nature 163:688

Roth, D. S., I. Perfecto, and B. Rathcke (1994) The effects of management systems on ground-foraging ant diversity in Costa Rica. Ecological Applications 4(3):423-436.

Examples

```
# Using Costa Rican ant diversity data from Roth et al. (1994)
data(ants)

# Calculate diversity for each covertype ("col")
shannons(ants[,2:ncol(ants)], ens = TRUE, counts = FALSE, margin = "col")

# Calculate diversity for each species ("row")
ant.div <- shannons(ants[,2:ncol(ants)], ens = TRUE, counts = FALSE,
                    margin = "row")
  row.names(ant.div) <- ants[,1]
  ant.div
```

`shift` *shift*

Description

Shift a vector by specified positive or negative lag

Usage

```
shift(x, lag = 1, pad = NA)
```

Arguments

`x`	A vector
`lag`	Number of lagged offsets, default is 1
`pad`	Value to fill the lagged offset with, default is NA

Value

A vector, length equal to x, with offset length filled with pad values

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
x <- 1:10

shift(x, 1)     # shift positive (from beginning of vector) by 1
shift(x, -1)    # shift negative (from end of vector) by 1
shift(x, 5, 0)  # Shift by 5 and fill (pad) with 0
```

`sieve` *Sieve raster data*

Description

Removes contiguous cells < specified query area

Usage

```
sieve(x, a, unit = c("m", "km", "ha"))
```

Arguments

`x`	An integer terra SpatRaster
`a`	Query area to remove
`unit`	The unit to use for area query options are c("m", "km", "ha")

Details

A sieve can be used to establish a minimal mapping unit where contiguous cells < specified query area are set to NA. These NA values can then be filled using focal (majority, median, mean)

Value

A terra SpatRaster with cells < a set to NA

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
library(terra)
elev <- rast(system.file("extdata/elev.tif", package="spatialEco"))
m <- matrix(c(100,200,1,200,300,2,300,400,3,400,
       500,4, 500,600,5), ncol=3, byrow=TRUE)
  x <- classify(elev, m)

# Sieve to a MMU of 60km
sv <- spatialEco::sieve(x, a = 60, unit = "km")
  plot(c(x, sv))
```

`similarity` *Ecological similarity*

Description

Uses row imputation to identify "k" ecological similar observations

Usage

```
similarity(
  x,
  k = 4,
  method = "mahalanobis",
  frequency = TRUE,
  scale = TRUE,
  ID = NULL
)
```

Arguments

`x`	data.frame containing ecological measures
`k`	Number of k nearest neighbors (kNN)
`method`	Method to compute multivariate distances c("mahalanobis", "raw", "euclidean", "ica")
`frequency`	Calculate frequency of each reference row (TRUE/FALSE)
`scale`	Scale multivariate distances to standard range (TRUE/FALSE)
`ID`	Unique ID vector to use as reference ID's (rownames). Must be unique and same length as number of rows in x

Details

This function uses row-based imputation to identify k similar neighbors for each observation. Has been used to identify offsets based on ecological similarity.

Value

data.frame with k similar targets and associated distances. If frequency = TRUE the freq column represents the number of times a row (ID) was selected as a neighbor.

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

References

Evans, J.S., S.R. Schill, G.T. Raber (2015) A Systematic Framework for Spatial Conservation Planning and Ecological Priority Design in St. Lucia, Eastern Caribbean. Chapter 26 in Central American Biodiversity : Conservation, Ecology and a Sustainable Future. F. Huettman (eds). Springer, NY.

Examples

```
library(sf)
data(pu)
kNN <- similarity(st_drop_geometry(pu[2:ncol(pu)]), k = 4,
                  frequency = TRUE, ID = pu$UNIT_ID)
p <- kNN$freq
clr <- c("#3288BD", "#99D594", "#E6F598", "#FEE08B",
       "#FC8D59", "#D53E4F")
p <- ifelse(p <= 0, clr[1],
       ifelse(p > 0 & p < 10, clr[2],
         ifelse(p >= 10 & p < 20, clr[3],
        ifelse(p >= 20 & p < 50, clr[4],
          ifelse(p >= 50 & p < 100, clr[5],
            ifelse(p >= 100, clr[6], NA))))))
plot(st_geometry(pu), col=p, border=NA)
  legend("topleft", legend=c("None","<10","10-20",
         "20-50","50-100",">100"),
         fill=clr, cex=0.6, bty="n")
  box()
```

`smooth.time.series` *Smooth Raster Time-series*

Description

Smooths pixel-level data in raster time-series and can impute missing (NA) values.

Usage

```
smooth.time.series(x, f = 0.8, smooth.data = FALSE, ...)
```

Arguments

`x`	A terra SpatRaster with > 8 layers
`f`	Smoothing parameter (see loess span argument)
`smooth.data`	(FALSE/TRUE) Smooth all of the data or just impute NA values
`...`	Additional arguments passed to terra::app (for writing results to disk)

Details

This function uses a LOESS regression to smooth the time-series. If the data is smoothed, (using the smooth.data = TRUE argument) it will be entirely replaced by a loess estimate of the time-series (estimated distribution at the pixel-level). Alternately, with smooth.data = FALSE, the function can be used to impute missing pixel data (NA) in raster time-series (stacks/bricks). The results can dramatically be effected by the choice of the smoothing parameter (f) so caution is warranted and the effect of this parameter tested.

Value

A terra SpatRaster containing imputed or smoothed data.

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

See Also

`loess` for details on the loess regression

`app` for details on additional (...) arguments

`impute.loess` for details on imputation model

Examples

```
library(terra)
  random.raster <- function(rows=50, cols=50, l=20, min=0, max=1){
    do.call(c, replicate(l, rast(matrix(runif(rows * cols, min, max),
        rows , cols))))
  }
r <- random.raster()

#### Smooth time-series using raster stack/brick
r.smooth <- smooth.time.series(r, f = 0.4, smooth.data = TRUE)

# extract pixel 100 for plotting
y <- as.numeric(r[100])
ys <- as.numeric(r.smooth[100])

# plot results
plot(y, type="l")
  lines(ys, col="red")
```

```
legend("bottomright", legend=c("original","smoothed"),
     lty=c(1,1), col=c("black","red"))
```

`sobal` *Sobel-Feldman operator*

Description

An isotropic image gradient operator using a 3x3 window

Usage

```
sobal(x, method = "intensity", ...)
```

Arguments

`x`	A raster class object
`method`	Type of operator ("intensity", "direction", "edge")
`...`	Additional arguments passed to raster::overlay or, if method="edge", raster::focal (if you want a file written to disk use filename = "" argument)

Details

The Sobel-Feldmanh operator is a discrete differentiation operator, deriving an approximation of the gradient of the intensity function. abrupt discontinuity in the gradient function represents edges, making this a common approach for edge detection. The Sobel-Feldman operator is based on convolving the image with a small, separable, and integer matrix in the horizontal and vertical directions. The operator uses two 3x3 kernels which are convolved with the original image to calculate approximations of the derivatives - one for horizontal changes, and one for vertical. Where x is defined here as increasing in the right-direction, and y as increasing in the down-direction. At each pixel in the raster, the resulting gradient can be combined to give the gradient intensity, using: SQRT(Gx^2 Gy^2). This can be expanded into the gradient direction using atan(Gx/Gy)

Value

A raster class object or raster written to disk

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

References

Sobel, I., & G. Feldman, (1969) A 3x3 Isotropic Gradient Operator for Image Processing, presented at the Stanford Artificial Intelligence Project (SAIL).

Examples

```
library(terra)

r <- rast(system.file("ex/logo.tif", package="terra"))
  s.int <- sobal(r[[1]])
  s.dir <- sobal(r[[1]], method = "direction")
  s.edge <- sobal(r[[1]], method = "edge")

opar <- par(no.readonly=TRUE)
par(mfrow=c(2,2))
  plot(r[[1]])
  plot(s.int, main="intensity")
  plot(s.dir, main="direction")
  plot(s.edge, main="edge")
par(opar)
```

`spatial.select` *Spatial Select*

Description

Performs a spatial select (feature subset) between a polygon(s) and other feature class

Usage

```
spatial.select(
  x,
  y = NULL,
  distance = NULL,
  predicate = c("intersection", "intersect", "contains", "covers", "touches",
    "proximity", "contingency"),
  neighbors = c("queen", "rook")
)
```

Arguments

`x`	An sp or sf polygon(s) object that defines the spatial query
`y`	A sp or sf feature class that will be subset by the query of x
`distance`	A proximity distance of features to select (within distance)
`predicate`	Spatial predicate for intersection
`neighbors`	If predicate = "contingency" type of neighbors options are c("queen", "rook")

Details

Performs a spatial select of features based on an overlay of a polygon (x), which can represent multiple features, and a polygon, point or line feature classes (y). User can specify a partial or complete intersection, using within argument, or within a distance, using distance argument, predicated on the query polygon. This function is similar to ArcGIS/Pro spatial select. Please note that for point to point neighbor selections use the knn function. Valid spatial predicates include: intersect, touches, covers, contains, proximity and contingency. See DE-9IM topology model for detailed information on following data predicates.

- intersection Create a spatial intersection between two features
- intersect Boolean evaluation of features intersecting
- contains Boolean evaluation of x containing y
- covers Boolean evaluation of x covering y
- touches Boolean evaluation of x touching y
- proximity Evaluation of distance-based proximity of x to y (x and y can be the same)
- contingency Evaluation of polygon contingency (eg., 1st, 2nd order)

Value

An sf object representing a subset of y based on the spatial query of x or, if predicate = contingency a sparse matrix representing neighbor indexes

Author(s)

Jeffrey S. Evans jeffrey_evans@tnc.org

See Also

`st_intersection` for details on intersection predicate

`st_intersects` for details on intersect predicate

`st_contains` for details on contain predicate

`st_covers` for details on covers predicate

`st_touches` for details on touches predicate

`st_is_within_distance` for details on proximity predicate

`https://en.wikipedia.org/wiki/DE-9IM` for details on DE-9IM topology model

Examples

```
if(require(sp, quietly = TRUE)) {
library(sf)
  data(meuse, package = "sp")
  meuse <- st_as_sf(meuse, coords = c("x", "y"), crs = 28992,
                    agr = "constant")

spolys <- hexagons(meuse, res=100)
  spolys$ID <- 1:nrow(spolys)
```

```
    p <- st_as_sf(st_sample(spolys, 500))
  p$PTID <- 1:nrow(p)
  sf::st_geometry(p) <- "geometry"

  plot(st_geometry(spolys), main="all data")
    plot(st_geometry(p), pch=20, add=TRUE)

sub.int <- spatial.select(p, spolys, predicate = "intersect")
  plot(st_geometry(sub.int), main="intersects")
    plot(st_geometry(p), pch=20, add=TRUE)

sub.prox <- spatial.select(p, spolys, distance=100, predicate = "proximity")
  plot(st_geometry(sub.int), main="intersects")
    plot(st_geometry(p), pch=20, add=TRUE)

# For rook or queen polygon contingency
plot( spolys <- sf::st_make_grid(sf::st_sfc(sf::st_point(c(0,0)),
                 sf::st_point(c(3,3))), n = c(3,3)) )

spatial.select(x=spolys, predicate = "contingency")
spatial.select(spolys, predicate = "contingency", neighbors = "rook")

} else {
  cat("Please install sp package to run example", "\n")
}
```

`spatialEcoNews` *spatialEco news*

Description

Displays release notes

Usage

```
spatialEcoNews(...)
```

Arguments

`...`	not used

Value

Shows package NEWS file

`spectral.separability` *spectral separability*

Description

Calculates spectral separability by class

Usage

```
spectral.separability(x, y, jeffries.matusita = TRUE)
```

Arguments

`x`	data.frame, matrix or vector of spectral values must, match classes defined in y
`y`	A vector or factor with grouping classes, must match row wise values in x
`jeffries.matusita`	(TRUE/FALSE) Return J-M distance (default) else Bhattacharyya

Details

Available statistics:

- Bhattacharyya distance (Bhattacharyya 1943; Harold 2003) measures the similarity of two discrete or continuous probability distributions.
- Jeffries-Matusita (default) distance (Bruzzone et al., 2005; Swain et al., 1971) is a scaled (0-2) version of Bhattacharyya. The J-M distance is asymptotic to 2, where 2 suggest complete separability.

Value

A matrix of class-wise Jeffries-Matusita or Bhattacharyya distance separability values

Author(s)

Jeffrey S. Evans jeffrey_evans@tnc.org

References

Bhattacharyya, A. (1943) On a measure of divergence between two statistical populations defined by their probability distributions'. Bulletin of the Calcutta Mathematical Society 35:99-109

Bruzzone, L., F. Roli, S.B. Serpico (1995) An extension to multiclass cases of the Jefferys-Matusita distance. IEEE Transactions on Pattern Analysis and Machine Intelligence 33:1318-1321

Kailath, T., (1967) The Divergence and Bhattacharyya measures in signal selection. IEEE Transactions on Communication Theory 15:52-60

Examples

```
require(MASS)
# Create example data
d <- 6                 # Number of bands
n.class <- 5           # Number of classes
n <- rep(1000, 5)
mu <- round(matrix(rnorm(d*n.class, 128, 1),
            ncol=n.class, byrow=TRUE), 0)

x <- matrix(double(), ncol=d, nrow=0)
  classes <- integer()
    for (i in 1:n.class) {
      f <- svd(matrix(rnorm(d^2), ncol=d))
      sigma <- t(f$v) %*% diag(rep(10, d)) %*% f$v
      x <- rbind(x, mvrnorm(n[i], mu[, i], sigma))
      classes <- c(classes, rep(i, n[i]))
    }
colnames(x) <- paste0("band", 1:6)
classes <- factor(classes, labels=c("water", "forest",
                  "shrub", "urban", "ag"))

# Separability for multi-band (multivariate) spectra
spectral.separability(x, classes)

# Separability for single-band (univariate) spectra
spectral.separability(x[,1], classes)
```

`spherical.sd` *Spherical Variance or Standard Deviation of Surface*

Description

Derives the spherical standard deviation of a raster surface

Usage

```
spherical.sd(r, d, variance = FALSE, ...)
```

Arguments

`r`	A terra SpatRaster class object
`d`	Size of focal window or a matrix to use in focal function
`variance`	(FALSE|TRUE) Output spherical variance rather than standard deviation
`...`	Additional arguments passed to terra:app (can write raster to disk here)

Details

Surface variability using spherical variance/standard deviation. The variation can be assessed using the spherical standard deviation of the normal direction within a local neighborhood. This is found by expressing the normal directions on the surfaces cells in terms of their displacements in a Cartesian (x,y,z) coordinate system. Averaging the x-coordinates, y-coordinates, and z-coordinates separately gives a vector (xb, yb, zb) pointing in the direction of the average normal. This vector will be shorter when there is more variation of the normals and it will be longest–equal to unity– when there is no variation. Its squared length is (by the Pythagorean theorem) given by: R^2 = xb^2 + yb^2 + zb^2 where; x = cos(aspect) * sin(slope) and xb = nXn focal mean of x y = sin(aspect) * sin(slope) and yb = nXn focal mean of y z = cos(slope) and zb = nXn focal mean of z

The slope and aspect values are expected to be in radians. The value of (1 - R^2), which will lie between 0 and 1, is the spherical variance. and it’s square root can be considered the spherical standard deviation.

Value

A terra SpatRaster class object of the spherical standard deviation

Author(s)

Jeffrey S. Evans <jeffrey_evans<at>tnc.org>

See Also

app for details on ... arguments

Examples

```
library(terra)
elev <- rast(system.file("extdata/elev.tif", package="spatialEco"))

ssd <- spherical.sd(elev, d=5)

slope <- terrain(elev, v='slope')
aspect <- terrain(elev, v='aspect')
hill <- shade(slope, aspect, 40, 270)
plot(hill, col=grey(0:100/100), legend=FALSE,
     main='terrain spherical standard deviation')
  plot(ssd, col=rainbow(25, alpha=0.35), add=TRUE)
```

`squareBuffer` *Square buffer*

Description

Creates a square buffer of a feature class

Usage

```
squareBuffer(x, a, ...)
```

Arguments

`x`	An sf object
`a`	Numeric single or vector indicating buffer distance(s)
`...`	Additional arguments passed to st_buffer

Details

Function creates a square buffer of feature class.

Value

A single feature sf class polygon object

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
library(sf)
xy <- st_as_sf(data.frame(x = c(1,3,6,7),
               y = c(3,2,7,8), z = c(38,23,12,12),
               area = c(32,23,45,67)),
      coords = c("x", "y"),
      agr = "constant")

# With fixed buffer
sb <- squareBuffer(xy, 32)
  plot(st_geometry(sb))
    plot(st_geometry(xy), pch=20, add=TRUE)

# With variable buffer
sb.var <- squareBuffer(xy, xy$area)
  plot(st_geometry(sb.var))
    plot(st_geometry(xy), pch=20, add=TRUE)
```

`srr` *Surface Relief Ratio*

Description

Calculates the Pike (1971) Surface Relief Ratio

Usage

```
srr(x, s = 5, ...)
```

Arguments

`x`	A terra SpatRaster object
`s`	Focal window size
`...`	Additional arguments passed to terra::lapp

Details

Describes rugosity in continuous raster surface within a specified window. The implementation of SRR can be shown as: (mean(x) - min(x)) / (max(x) - min(x))

Value

A terra SpatRaster object of Pike's (1971) Surface Relief Ratio

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
library(terra)
elev <- rast(system.file("extdata/elev.tif", package="spatialEco"))
  r.srr <- srr(elev, s=5)
    plot(r.srr, main="Surface Relief Ratio")
```

`stratified.random` *Stratified random sample*

Description

Creates a stratified random sample of an sp class object

Usage

```
stratified.random(x, strata, n = 10, reps = 1, replace = FALSE)
```

Arguments

`x`	An sf class object
`strata`	Column in x with stratification factor
`n`	Number of random samples
`reps`	Number of replicates per strata
`replace`	(TRUE/FALSE) Sampling with replacement

Details

If replace=FALSE features are removed from consideration in subsequent replicates. Conversely, if replace=TRUE, a feature can be selected multiple times across replicates. Not applicable if rep=1.

Value

An sf class object containing random samples

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

References

Hudak, A.T., N.L. Crookston, J.S. Evans, M.J. Falkowski, A.M.S. Smith, P. Gessler and P. Morgan. (2006) Regression modelling and mapping of coniferous forest basal area and tree density from discrete-return lidar and multispectral satellite data. Canadian Journal of Remote Sensing 32: 126-138.

Examples

```
if(require(sp, quietly = TRUE)) {
library(sf)
  data(meuse, package = "sp")
  meuse <- st_as_sf(meuse, coords = c("x", "y"), crs = 28992,
                    agr = "constant")
```

```
# Create stratified variable using quartile breaks
x1 <- cut(meuse$cadmium, summary(meuse$cadmium)[-4],
          include.lowest=TRUE)
  levels(x1) <- seq(1,nlevels(x1),1)
x2 <- cut(meuse$lead, summary(meuse$lead)[-4],
          include.lowest=TRUE)
  levels(x2) <- seq(1,nlevels(x2),1)
meuse$STRAT <- paste(x1, x2, sep='.')

# Counts for each full strata (note; 2 strata have only 1 observation)
tapply(meuse$STRAT, meuse$STRAT, length)

# 2 replicates, 2 samples with replacement
ssample <- stratified.random(meuse, strata='STRAT', n=2, reps=2,
                             replace=TRUE)
  tapply(ssample$STRAT, ssample$STRAT, length)

# 2 replicates, 2 samples no replacement
ssample.nr <- stratified.random(meuse, strata='STRAT', n=2, reps=2)
  tapply(ssample.nr$STRAT, ssample.nr$STRAT, length)

# n=1 and reps=10 for sequential numbering of samples
ssample.ct <- stratified.random(meuse, strata='STRAT', n=1, reps=10,
                                replace=TRUE)
  tapply(ssample.ct$STRAT, ssample.ct$STRAT, length)

# Plot random samples colored by replacement
ssample$REP <- factor(ssample$REP)
  plot(ssample['REP'], pch=20)

} else {
  cat("Please install sp package to run example", "\n")
}
```

`subsample.distance` *Distance-based subsampling*

Description

Draws a minimum, and optional maximum constrained, distance sub-sampling

Usage

```
subsample.distance(x, size, d, d.max = NULL, replacement = FALSE)
```

Arguments

`x`	A POLYGON or POINT sf object
`size`	Subsample size

`d`	Minimum sampling distance in meters
`d.max`	Maximum sampling distance in meters
`replacement`	(FALSE/TRUE) Subsample with replacement

Value

A subsampled POLYGON or POINT sf object

Note

This function provides a distance constrained subsample of existing point or polygon data. Please note that units are in meters regardless of input CRS projection units (including lat/long).

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
if(require(sp, quietly = TRUE)) {
library(sf)
  data(meuse, package = "sp")
  meuse <- st_as_sf(meuse, coords = c("x", "y"), crs = 28992,
                    agr = "constant")

# Subsample with a 500m minimum sample spread
sub.meuse <- subsample.distance(meuse, size = 10, d = 500)
  plot(st_geometry(meuse), pch=19, main="min dist = 500")
    plot(st_geometry(sub.meuse), pch=19, col="red", add=TRUE)

# Check distances
dm <- st_distance(sub.meuse)
  diag(dm) <- NA
cat("\n", "Min distance for subsample", min(dm, na.rm=TRUE), "\n")
cat("Max distance for subsample", max(dm, na.rm=TRUE), "\n")

} else {
  cat("Please install sp package to run example", "\n")
}
```

`summary.cross.cor` *Summary of spatial cross correlation*

Description

summary method for class "cross.cor"

Usage

```
## S3 method for class 'cross.cor'
summary(object, ...)
```

Arguments

`object`	Object of class cross.cor
`...`	Ignored

Value

When not simulated k=0, prints functions list object containing:

- I - Global autocorrelation statistic
- SCI - - A data.frame with two columns representing the xy and yx autocorrelation
- nsim - value of NULL to represent p values were derived from observed data (k=0)
- p - Probability based observations above/below confidence interval
- t.test - Probability based on t-test

When simulated (k>0), prints functions list object containing:

- I - Global autocorrelation statistic
- SCI - A data.frame with two columns representing the xy and yx autocorrelation
- nsim - value representing number of simulations
- global.p - p-value of global autocorrelation statistic
- local.p - Probability based simulated data using successful rejection of t-test
- range.p - Probability based on range of probabilities resulting from paired t-test

---

`summary.effect.size` *Summarizing effect size*

Description

Summary method for class "effect.size".

Usage

```
## S3 method for class 'effect.size'
summary(object, ...)
```

Arguments

`object`	Object of class effect.size
`...`	Ignored

Value

Prints the output data.frame contaning; effect size with upper and lower confidence and, mean and sd by group

---

`summary.loess.boot` *Summarizing Loess bootstrap models*

Description

Summary method for class "loess.boot".

Usage

```
## S3 method for class 'loess.boot'
summary(object, ...)
```

Arguments

`object`	Object of class loess.boot
`...`	Ignored

Value

same as print lowess.boot data.frame including;

- nreps Number of bootstrap replicates
- confidence Confidence interval (region)
- span alpha (span) parameter used loess fit
- degree polynomial degree used in loess fit
- normalize Normalized data (TRUE/FALSE)
- family Family of statistic used in fit
- parametric Parametric approximation (TRUE/FALSE)
- surface Surface fit, see loess.control
- data data.frame of x,y used in model
- fit data.frame including:
  1. x - Equally-spaced x index (see NOTES)
  2. y.fit - loess fit
  3. up.lim - Upper confidence interval
  4. low.lim - Lower confidence interval
  5. stddev - Standard deviation of loess fit at each x value

---

`swvi` *Senescence weighted Vegetation Index (swvi)*

Description

Modified Soil-adjusted Vegetation Index (MSAVI) or Modified Triangular Vegetation Index 2 (MTVI) weighted by the Normalized difference senescent vegetation index (NDSVI)

Usage

```
swvi(
  red,
  nir,
  swir,
  green = NULL,
  mtvi = FALSE,
  senescence = 0,
  threshold = NULL,
  weight.factor = NULL,
  ...
)
```

Arguments

`red`	SpatRaster, Red band (0.636 - 0.673mm), landsat 5&7 band 3, OLI (landsat 8) band 4
`nir`	SpatRaster, Near infrared band (0.851 - 0.879mm) landsat 5&7 band 4, OLI (landsat 8) band 5
`swir`	SpatRaster, short-wave infrared band 1 (1.566 - 1.651mm), landsat 5&7 band 5, OLI (landsat 8) band 6
`green`	Green band if MTVI = TRUE
`mtvi`	(FALSE | TRUE) Use Modified Triangular Vegetation Index 2 instead of MSAVI
`senescence`	The critical value, in NDSVI, representing senescent vegetation
`threshold`	Threshold value for defining NA based on < p
`weight.factor`	Apply partial weights (w * weight.factor) to the NDSVI weights
`...`	Additional arguments passed to terra::lapp function

Details

The intent of this index is to correct the MSAVI or MTVI index for bias associated with senescent vegetation. This is done by: 1 deriving the NDSVI 2 applying a threshold to limit NDSVI to values associated with senescent vegetation 3 converting the index to inverted weights (-1*(NDSVI/sum(NDSVI))) 4 applying weights to MSAVI or MTVI

The MSAVI formula follows the modification proposed by Qi et al. (1994), often referred to as MSAVI2. MSAVI index reduces soil noise and increases the dynamic range of the vegetation signal. The implemented modified version (MSAVI2) is based on an inductive method that does not use a constant L value, in separating soil effects, an highlights healthy vegetation. The MTVI(2) index follows Haboudane et al., (2004) and represents the area of a hypothetical triangle in spectral space that connects (1) green peak reflectance, (2) minimum chlorophyll absorption, and (3) the NIR shoulder. When chlorophyll absorption causes a decrease of red reflectance, and leaf tissue abundance causes an increase in NIR reflectance, the total area of the triangle increases. It is good for estimating green LAI, but its sensitivity to chlorophyll increases with an increase in canopy density. The modified version of the index accounts for the background signature of soils while preserving sensitivity to LAI and resistance to the influence of chlorophyll.

The Normalized difference senescent vegetation index (NDSVI) follows methods from Qi et a., (2000). The senescence is used to threshold the NDSVI. Values less then this value will be NA. The threshold argument is used to apply a threshold to MSAVI. The default is NULL but if specified all values (MSAVI <= threshold) will be NA. Applying a weight.factor can be used to change the influence of the weights on MSAVI.

Value

A terra SpatRaster class object of the weighted MSAVI metric

Author(s)

Jeffrey S. Evans jeffrey_evans@tnc.org

References

Haboudane, D., et al. (2004) Hyperspectral Vegetation Indices and Novel Algorithms for Predicting Green LAI of Crop Canopies: Modeling and Validation in the Context of Precision Agriculture. Remote Sensing of Environment 90:337-352.

Qi J., Chehbouni A., Huete A.R., Kerr Y.H., (1994). Modified Soil Adjusted Vegetation Index (MSAVI). Remote Sens Environ 48:119-126.

Qi J., Kerr Y., Chehbouni A., (1994). External factor consideration in vegetation index development. Proc. of Physical Measurements and Signatures in Remote Sensing, ISPRS, 723-730.

Qi, J., Marsett, R., Moran, M.S., Goodrich, D.C., Heilman, P., Kerr, Y.H., Dedieu, G., Chehbouni, A., Zhang, X.X. (2000). Spatial and temporal dynamics of vegetation

Examples

```
library(terra)
lsat <- rast(system.file("/extdata/Landsat_TM5.tif", package="spatialEco"))

# Using Modified Soil-adjusted Vegetation Index (MSAVI)
( wmsavi <- swvi(red = lsat[[3]], nir = lsat[[4]], swir = lsat[[5]]) )
    plotRGB(lsat, r=6,g=5,b=2, scale=1, stretch="lin")
      plot(wmsavi, legend=FALSE, col=rev(terrain.colors(100, alpha=0.35)), add=TRUE )

# Using Modified Triangular Vegetation Index 2 (MTVI)
```

```
( wmtvi <- swvi(red = lsat[[3]], nir = lsat[[4]], swir = lsat[[5]],
                          green = lsat[[3]], mtvi = TRUE) )
  plotRGB(lsat, r=6,g=5,b=2, scale=1, stretch="lin")
    plot(wmtvi, legend=FALSE, col=rev(terrain.colors(100, alpha=0.35)), add=TRUE )
```

`time_to_event` *Time to event*

Description

Returns the time (sum to position) to a specified value

Usage

```
time_to_event(
  x,
  y = 1,
  dir = c("LR", "RL"),
  int = FALSE,
  up.to = FALSE,
  na.action = c("fail", "ignore")
)
```

Arguments

`x`	A vector, representing time-series, to evaluate
`y`	Threshold value tor return position for
`dir`	Direction of evaluation c("LR", "RL")
`int`	FALSE | TRUE - Evaluate as integer (rounds to 0 decimal places)
`up.to`	FALSE | TRUE - Return value before event
`na.action`	c("fail", "ignore"), if "fail" function will return error with NA's with "ignore" NA values will be included in count to event

Details

The time to event represents the sum of positions, in the vector, until the specified value is found ie., (0,0,1) would be 3 or, 2 with up.to=TRUE. The int argument allows for rounding a continuous variable. Since it may be difficult to find an exact match to a floating point value rounding mitigates the problem. If you want a specific rounding value (eg., 1 decimal place) you can apply it to x first then pass it to the function. The up.to argument will stop one value before the specified value of (y) regardless of integer or float. For NA handling, na.action defines the function behavior, causing it to fail or count NAs. Note that it makes no sense to actually remove NAs as it will make the run uninterpretable.

Value

A vector value representing the time to event

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
# Binomial instance
time_to_event(c(0,0,0,0,1,0,0,0,1,0))
time_to_event(c(0,0,0,0,1,0,0,0,1,0), up.to = TRUE)
time_to_event(c(0,0,0,0,1,0,0,0,1,0), dir="RL")
time_to_event(c(NA,0,0,0,1,0,0,0,1,0), na.action="ignore")

# Continuous threshold instance
( x <- runif(100, 0,7) )
time_to_event(x, y = 5, int=TRUE)

# raster example
library(terra)

# Binomial instance
r <- do.call(c, replicate(20,terra::rast(matrix(sample(
             c(0,1), 1000, replace=TRUE), 100, 100))))
  ( t2e <- app(r, fun=time_to_event) )

# Continuous threshold instance
r <- do.call(c, replicate(20,terra::rast(matrix(
              runif(1000,0,7), 100, 100))))
  ( t2e <- app(r, fun=time_to_event, y=5) )
```

`TM5` *Landsat 5 TM Scene*

Description

Subset of Landsat 5 TM Scene: LT52240631988227CUB02 Contains six bands of surface reflectance path 224/row 63 acquisition date: 1988-08-14 13:00:47 GMT, EPSG:32622

Format

A tif (inst/extdata/Landsat_TM5.tif) with 30m 6 bands:

Blue 0.45 - 0.52 µm

Green 0.52 - 0.60 µm

Red 0.63 - 0.69 µm

NIR Near-Infrared 0.76 - 0.90 µm

SWIR1 Near-Infrared 1.55 - 1.75 µm

SWIR2 Mid-Infrared 2.08 - 2.35 µm

`topo.distance` *Topographic distance*

Description

Calculates topographic corrected distance for a line object

Usage

```
topo.distance(x, r, echo = FALSE)
```

Arguments

`x`	sf LINESTRING object
`r`	terra SpatRaster class elevation raster
`echo`	(FALSE/TRUE) print progress to screen

Details

This function corrects straight-line (euclidean) distances for topographic-slope effect.

Value

Vector of corrected topographic distances same length as nrow(x)

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
library(sf)
library(terra)

# create example data
elev <- rast(system.file("extdata/elev.tif", package="spatialEco"))
  names(elev) <- "elev"

lns <- lapply(1:5, function(i) {
  p <- st_combine(st_as_sf(spatSample(elev, size=2, as.points=TRUE)))
  st_as_sf(st_cast(p, "LINESTRING")) })
lns <- do.call(rbind, lns)

 plot(elev)
```

```
    plot(st_geometry(lns), add=TRUE)

# Calculate topographical distance
( tdist <- topo.distance(lns, elev) )
( lgt <- as.numeric(st_length(lns)) )

# Increase in corrected distance
tdist - lgt

# Percent increase in corrected distance
((tdist - lgt) / lgt) * 100
```

`tpi` *Topographic Position Index (tpi)*

Description

Calculates topographic position using mean deviations

Usage

```
tpi(x, scale = 3, win = "rectangle", normalize = FALSE, zero.correct = FALSE)
```

Arguments

`x`	A terra SpatRaster object
`scale`	focal window size (n-cell x n-cell for rectangle or distance for circle)
`win`	Window type. Options are "rectangle" and "circle"
`normalize`	Apply deviation correction that normalizes to local surface roughness
`zero.correct`	Apply correction for zero values in matrix weights

Value

A terra SpatRaster object of tpi A terra SpatRaster object

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

References

De Reu, J., J. Bourgeois, M. Bats, A. Zwertvaegher, V. Gelorini, et al., (2014) Application of the topographic position index to heterogeneous landscapes. Geomorphology, 186:39-49.

Examples

```
library(terra)
elev <- rast(system.file("extdata/elev.tif", package="spatialEco"))

# calculate tpi and plot
  tpi7 <- tpi(elev, scale=7)
  tpi025 <- tpi(elev, win = "circle", scale=2500)
  tpi025.zc <- tpi(elev, win = "circle", scale=2500,
                   zero.correct = TRUE)

opar <- par(no.readonly=TRUE)
    par(mfrow=c(2,2))
      plot(elev, main="original raster")
      plot(tpi7, main="tpi 7x7")
      plot(tpi025, main="tpi Circular window d=2500m")
   plot(tpi025, main="tpi Circular window d=2500m, zero correct")
par(opar)
```

`trasp` *Solar-radiation Aspect Index*

Description

Calculates the Roberts and Cooper (1989) Solar-radiation Aspect Index

Usage

```
trasp(x, ...)
```

Arguments

`x`	A terra SpatRaster object
`...`	Additional arguments passed to terra::app

Details

Roberts and Cooper (1989) rotates (transforms) the circular aspect to assign a value of zero to land oriented in a north-northeast direction, (typically the coolest and wettest orientation), and a value of one on the hotter, dryer south-southwesterly slopes. The result is a continuous variable between 0 - 1. The metric is defined as: trasp = (1 - cos((pi/180)(a-30)) / 2 where; a = aspect in degrees

Value

A terra SpatRaster object of Roberts and Cooper (1989) Solar-radiation Aspect Index

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

References

Roberts. D.W., and Cooper, S.V. (1989). Concepts and techniques of vegetation mapping. In Land Classifications Based on Vegetation: Applications for Resource Management. USDA Forest Service GTR INT-257, Ogden, UT, pp 90-96

Examples

```
library(terra)
elev <- rast(system.file("extdata/elev.tif", package="spatialEco"))
  s <- trasp(elev)
    plot(s)
```

`trend.line` *trend.line*

Description

Calculated specified trend line of x,y

Usage

```
trend.line(x, y, type = "linear", plot = TRUE, ...)
```

Arguments

`x`	Vector of x
`y`	Vector of y
`type`	Trend line types are: 'linear', 'exponential', 'logarithmic', 'polynomial'
`plot`	plot results (TRUE/FALSE)
`...`	Additional arguments passed to plot

Value

A list class object with the following components:

- for type = 'linear' x is slope and y is intercept
- for type = 'exponential', 'logarithmic', or 'polynomial' x is original x variable and y is vector of fit regression line

Author(s)

Jeffrey S. Evans jeffrey_evans@tnc.org

Examples

```
x <- 1:10
y <- jitter(x^2)

opar <- par(no.readonly=TRUE)
  par(mfcol=c(2,2))
    trend.line(x,y,type='linear',plot=TRUE,pch=20,main='Linear')
    trend.line(x,y,type='exponential',plot=TRUE,pch=20,main='Exponential')
    trend.line(x,y,type='logarithmic',plot=TRUE,pch=20,main='Logarithmic')
    trend.line(x,y,type='polynomial',plot=TRUE,pch=20,main='Polynomial')
 par(opar)
```

`tri` *Terrain Ruggedness Index*

Description

Implementation of the Riley et al (1999) Terrain Ruggedness Index

Usage

```
tri(r, s = 3, exact = TRUE, ...)
```

Arguments

`r`	A terra SpatRaster class object
`s`	Scale of window. Must be odd number, can represent 2 dimensions (eg., s=c(3,5) would represent a 3 x 5 window)
`exact`	Calculate (TRUE/FALSE) the exact TRI or an algebraic approximation.
`...`	Additional arguments passed to terra::focal or terra::app

Details

The algebraic approximation is considerably faster. However, because inclusion of the center cell, the larger the scale the larger the divergence of the minimum value. Resuls are driven by local variations so, fixed thresholds are not very reliable. However there are some reccomended breaks (eg., Riley et al., 1999).

Riley et al., (1999) ranges for classifying Topographic Ruggedness Index:

- 0-80 - level terrain surface.
- 81-116 - nearly level surface.
- 117-161 - slightly rugged surface.
- 162-239 - intermediately rugged surface.
- 240-497 - moderately rugged surface.
- 498-958 - highly rugged surface.
- gt 959 - extremely rugged surface.

Value

A terra SpatRaster class object of the TRI

Author(s)

Jeffrey S. Evans jeffrey_evans@tnc.org

References

Riley, S.J., S.D. DeGloria and R. Elliot (1999) A terrain ruggedness index that quantifies topographic heterogeneity, Intermountain Journal of Sciences 5(1-4):23-27.

Examples

```
library(terra)
elev <- rast(system.file("extdata/elev.tif", package="spatialEco"))
  ( tri.ext <- tri(elev) )
  ( tri.app <- tri(elev, exact = FALSE) )
  plot(c(tri.ext, tri.app))
```

`vrm` *Vector Ruggedness Measure (VRM)*

Description

Implementation of the Sappington et al., (2007) vector ruggedness measure

Usage

```
vrm(x, s = 3)
```

Arguments

`x`	A terra SpatRaster class object
`s`	Scale of window. Must be odd number, can represent 2 dimensions (eg., s=c(3,5) would represent a 3 x 5 window)

Details

This function measures terrain ruggedness by calculating the vector ruggedness measure

Value

A terra SpatRaster class object of the VRI

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

References

Sappington, J.M., K.M. Longshore, D.B. Thomson (2007). Quantifying Landscape Ruggedness for Animal Habitat Analysis: A case Study Using Bighorn Sheep in the Mojave Desert. Journal of Wildlife Management. 71(5):1419-1426

Examples

```
library(terra)
elev <- rast(system.file("extdata/elev.tif", package="spatialEco"))
   vrm3 <- vrm(elev)
   vrm5 <- vrm(elev, s=5)
   plot(c(vrm3, vrm5))
```

`winsorize` *Winsorize transformation*

Description

Removes extreme outliers using a winsorization transformation

Usage

```
winsorize(
  x,
  min.value = NULL,
  max.value = NULL,
  p = c(0.05, 0.95),
  na.rm = FALSE
)
```

Arguments

`x`	A numeric vector
`min.value`	A fixed lower bounds, all values lower than this will be replaced by this value. The default is set to the 5th-quantile of x.
`max.value`	A fixed upper bounds, all values higher than this will be replaced by this value. The default is set to the 95th-quantile of x.
`p`	A numeric vector of 2 representing the probabilities used in the quantile function.
`na.rm`	(FALSE/TRUE) should NAs be omitted?

Details

Winsorization is the transformation of a distribution by limiting extreme values to reduce the effect of spurious outliers. This is done by shrinking outlying observations to the border of the main part of the distribution.

Value

A transformed vector the same length as x, unless na.rm is TRUE, then x is length minus number of NA's

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

References

Dixon, W.J. (1960) Simplified Estimation from Censored Normal Samples. Annals of Mathematical Statistics. 31(2):385-391

Examples

```
set.seed(1234)
x <- rnorm(100)
x[1] <- x[1] * 10
winsorize(x)

plot(x, type="l", main="Winsorization transformation")
  lines(winsorize(x), col="red", lwd=2)
    legend("bottomright", legend=c("Original distribution","With outliers removed"),
        lty=c(1,1), col=c("black","red"))

# Behavior with NA value(s)
x[4] <- NA
winsorize(x)             # returns x with original NA's
winsorize(x, na.rm=TRUE) # removes NA's
```

`wt.centroid` *Weighted centroid*

Description

Creates centroid of [x,y] coordinates with optional weights field

Usage

```
wt.centroid(x, p = NULL, spatial = TRUE)
```

Arguments

`x`	sf POINT class object
`p`	Weights column in x
`spatial`	(TRUE/FALSE) Output sf POINT object

Details

The weighted centroid is calculated as: [Xw]=[X]*[p], [Yw]=[Y]*[p], [sXw]=SUM[Xw], [sYw]=SUM[Yw], [sP]=SUM[p] wX=[sXw]/[sP], wY=[sYw]/[sP] where; X=X coordinate(S), Y=Y coordinate(S), p=WEIGHT

Value

An x,y coordinate or sf POINT object representing the weighted or unweighted coordinate centroid

Author(s)

Jeffrey S. Evans <jeffrey_evans@tnc.org>

Examples

```
p = c("sf", "sp")
if(any(!unlist(lapply(p, requireNamespace, quietly=TRUE)))) {
  m = which(!unlist(lapply(p, requireNamespace, quietly=TRUE)))
  message("Can't run examples, please install ", paste(p[m], collapse = " "))
} else {
invisible(lapply(p, require, character.only=TRUE))

data(meuse, package = "sp")
meuse <- st_as_sf(meuse, coords = c("x", "y"), crs = 28992,
                  agr = "constant")

wt.copper <- wt.centroid(meuse, p='copper')
wt.zinc <- wt.centroid(meuse,  p='zinc')

plot(st_geometry(meuse), pch=20, cex=0.75,
     main='Weighted centroid(s)')
  plot(st_geometry(wt.copper), pch=19, col='red',
       cex=1.5, add=TRUE)
  plot(st_geometry(wt.zinc), pch=19, col='blue',
       cex=1.5, add=TRUE)
legend('topleft', legend=c('all','copper', 'zinc'),
       pch=c(20,19,19),col=c('black','red','blue'))
}
```

Index