FUN_SPDF_Cmi_Index.R

duccioa

Thu Dec 24 18:23:12 2015

```
library(sp)

extract.areas <- function(sp){
    n <- length(sp@polygons)
    areas <- rep(0, n)
    for(i in 1:n){
        areas[i] <- sp@polygons[i][[1]]@Polygons[[1]]@area
    }
    return(areas)
}
#1#
#creates the matrix V with coordinates x1,y1,x2,y2 of each segment of the polygon
fetch.lines <- function(sp, i){
    coords <- sp@polygons[i][[1]]@Polygons[[1]]@coords#extract the coordinates of the vertex
    n <- nrow(coords)
    coords <- coords[n:1, ]
    v <- cbind(coords[1:(n-1),], coords[2:n,])
    return(v)
}
##########################################################################################
#2#
#Given one vector with coordinates x1,y1,x2,y2 the function return
#Areas and MI of the triangle and rectangle for the line between the two vertex
#v[1]=x1
#v[2]=y1
#v[3]=x2
#v[4]=y2
#To be applied to matrix V
#Caclulate area
calc.A_tr <- function(v){
    A_tr <- (v[3]-v[1])*(v[4]-v[2])/2#area of the triangle
    return(A_tr)
}
calc.A_rec <- function(v){
    A_rec <- (v[3]-v[1])*v[2]#area of the rectangle
    return(A_rec)
}
##########################################################################################
#3#
#calculate x_g and y_g coordinates
#specify the required coordinate "x" or "y"
#specify the shape "tri" or "rec"
calc.coords_g <- function(v, x_or_y = "x", shape = "tri"){
    if(x_or_y == "x"){
        if(shape == "tri"){return((v[1]+2*v[3])/3)}#centroid's x_gt of the triangle
        if(shape == "rec"){return((v[2]+2*v[4])/3)}#centroid's y_gt of the triangle
    }
```

```
    if(x_or_y == "y"){
        if(shape == "tri"){return((v[1]+v[3])/2)}#centroid's x_gr of the rectangle
        if(shape == "rec"){return(v[2]/2)}#centroid's y_gr of the rectangle
    }
}
##########################################################################################
#4#
#calculate MI
calc.I_tr <- function(v, Area){Area*((v[3]-v[1])^2 + (v[2]-v[4])^2)/18}#I_tr

calc.I_rec <- function(v, Area){Area*((v[3]-v[1])^2 + v[2])/12}#I_rec
##########################################################################################

#5#
#Create dataframe to df_MI used to perform the final calculation of the MIg
create.df <- function(V){
    A_tr <- apply(V, 1, calc.A_tr)#Area triangle
    A_rec <- apply(V, 1, calc.A_rec)#Area rectangle
    x_gt <- apply(V, 1, calc.coords_g, x_or_y = "x", shape = "tri")#x coords of the triangle's centroid
    y_gt <- apply(V, 1, calc.coords_g, x_or_y = "y", shape = "tri")
    x_gr <- apply(V, 1, calc.coords_g, x_or_y = "x", shape = "rec")
    y_gr <- apply(V, 1, calc.coords_g, x_or_y = "y", shape = "rec")
    I_tr <- calc.I_tr(V, Area = A_tr)#MI of the triangle about its centroid
    I_rec <- calc.I_rec(V, Area = A_rec)
    V_df <- data.frame(line_ID = seq(1, nrow(V)),
                       cbind(V, x_gt, y_gt, A_tr, I_tr, x_gr, y_gr, A_rec, I_rec))
    names(V_df)[2:5] <- c("x1", "y1", "x2", "y2")
    A <- sum(A_tr + A_rec)#total area triangle and rectangle
    x_g <- sum((x_gr*A_rec + x_gt*A_tr)/A)#x coords of the polygon's centroid
    y_g <- sum((y_gr*A_rec + y_gt*A_tr)/A)
    V_df$Dist_tr <- sqrt((x_g - x_gt)^2 + (y_g - y_gt)^2)#distance
    V_df$Dist_rec <- sqrt((x_g - x_gr)^2 + (y_g - y_gr)^2)
    return(V_df)
}
##########################################################################################
#6#
calc.Cmi_index <- function(V_df, A_pol){
    Ig <- with(V_df, sum(I_tr + A_tr*(Dist_tr^2) + I_rec + A_rec*(Dist_rec^2)))
    C_mi <- (A_pol^2)/(2*pi*Ig)
    return(C_mi)
}
##########################################################################################
##########################################################################################

Polygon.Cmi_index <- function(sp, i){
    Area_polygon <- sp@polygons[i][[1]]@Polygons[[1]]@area
    V_matrix <- fetch.lines(sp, i)
    V_polygon <- create.df(V_matrix)
    C_mi <- calc.Cmi_index(V_polygon, Area_polygon)
    return(C_mi)
}
```

```r
##Add a column of C_mi indexes to the polygon dataframe
SPDF.Cmi_Index <- function(spdf){
    n <- length(spdf@polygons)
    polygons.areas <- extract.areas(spdf)
    Cmi_indexes <- rep(0, n)
    for(i in 1:n){
        Cmi_indexes[i] <- Polygon.Cmi_index(spdf, i)
    }
    spdf@data$C_mi <- Cmi_indexes
    return(spdf)
}
```