About a local Duncan segregation index

Geoffrey Caruso

2023-12-14

Scope

This document showcases an implementation of a local version of the Duncan segregation index. In this local version, segregation is to be undesrtood within a given neighbourhood range, not over the entire space. Although there is no theoretical link drawn between the global and local versions (as would be the case for Moran's I for example), the local.duncan is an approach that seems to be able to stress spatial discontinuities in segregation, with some regions being highly segregated globally but having few spatial discontinuities or, conversely, many. It would probably need to be compared to location quotients, joint count statistics and other co-location indices, or even LISA's on group percentage, as well as to aggregate - but spatial - versions of Duncan that account for some spatial structure (see e.g. Apparicio, P: https://doi.org/10.4000/cybergeo.12063). As of today the metric also simply needs to be further digested to better grasp its properties.

This local.duncan was made from a demand to have a mapped version of a segregation index as part of the census 2021 analysis project in Luxembourg (involving the University of Luxembourg, LISER, and Statec (statistical office)) and conversation with Frederic Docquier, Philippe Gerber and Isabelle Pigeron-Piroth. Geoffrey Caruso suggested a "local" version of the Duncan, thinking it would already exist, but then couldn't find any. Geoffrey Caruso then made a quick terra-based version of a 'local.duncan' with Yann Ferro (who quickly found out doing it with arcgis was terribly inefficient).

We propose here a cleaner vector-based development using a neighbours list and simple feature access (sf) polygon support, leading to a main function named local.duncan(). Time permitting, the primary raster version will be properly reworked as a focal.duncan() function using the focal function in terra. A result of that first raster version is shown in the publication 9 at https://statistiques.public.lu/fr/recensement.html. And the code is at the moment available at https://github.com/geocaruso/cartolux/blob/eea3ac42f82eeaf2a29b8abd65606fba8cee4e6c/R/script__indice__duncan.R

Starting example - duncan()

A simple duncan() function is first created to be used within the next function. It is stored in "R/duncan.R". It is a simple implementation for a data.fram of the Duncan index with n units i and population types A and B:

$$\frac{1}{2}\sum_{i}^{n}\left|\frac{A_i}{\sum_i^n A_i}-\frac{B_i}{\sum_i^n B_i}\right|$$

```r
duncan<-function(df,PopA,PopB){
  A<-df[,PopA]
  B<-df[,PopB]
  a<-A/sum(A)
  b<-B/sum(B)
  diff_ab<-a-b
  D<-sum(abs(diff_ab))/2
  D}
```

Applied to a simple data.frame:

```r
df<-data.frame(ID=c("I","II","III","IV"),A=c(4,4,1,0), B=c(0,0,2,2), C=c(1,1,1,1))
list(duncan(df,"A","B"), duncan(df,"A","C"), duncan(df,"B","C"))
```

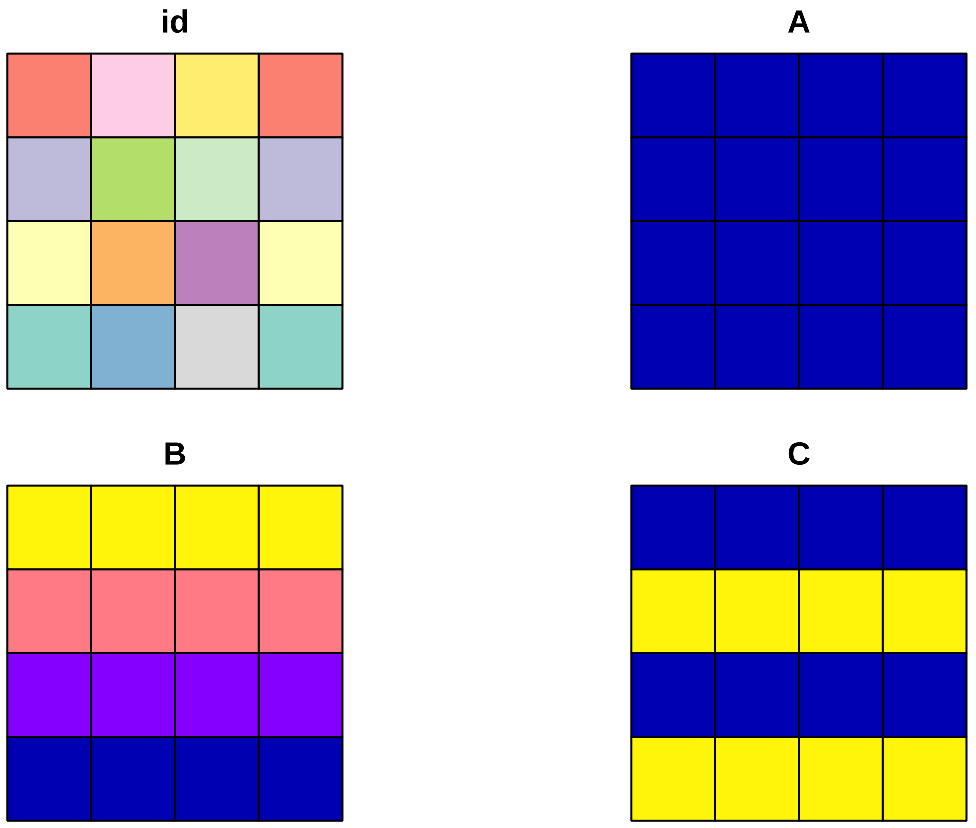
```
## [[1]]
## [1] 0.8888889
##
## [[2]]
## [1] 0.3888889
##
## [[3]]
## [1] 0.5
```

Showing the higher segregation of A and B population and, since B and C have same population that moving half of them would lead to a uniform distribution.

Second example with a polygon geography - global.duncan()

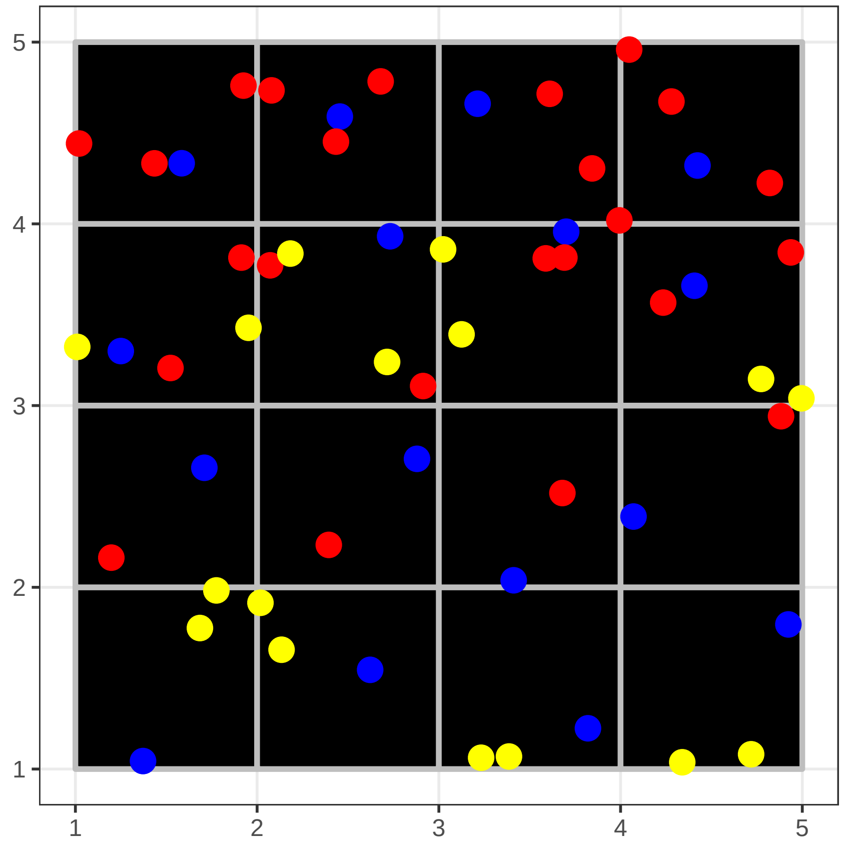
Let's first build a 4 x 4 squared geography and populate it with A (uniform), B and C

```r
x1<-rep((1:4),each=4)
y1<-rep((1:4),4)
df<-data.frame(id=paste(x1,y1,sep="_"))
df$geom<-sprintf("POLYGON((%s %s, %s %s, %s %s, %s %s, %s %s))",
                 x1, y1, x1, y1+1, x1+1, y1+1, x1+1, y1, x1, y1)
sf<-sf::st_as_sf(df, wkt = "geom")
#Populate
sf$A<-1
sf$B<-rep((0:3),4)
sf$C<-rep(c(2,0),8)
#Plot
plot(sf)
```



Or mapped the ggplot way and adding a visulaisation of the two groups randomly positioned

```r
library(ggplot2)
g<-ggplot()+
  geom_sf(data=sf, fill="black", col="grey", linewidth=1)+
  theme_bw()

set.seed(101)
Apt<-sf::st_sample(sf, size = sf$A)
Bpt<-sf::st_sample(sf, size = sf$B)
Cpt<-sf::st_sample(sf, size = sf$C)

blues<-geom_sf(data=Apt, size=4, col="blue")
reds<-geom_sf(data=Bpt, size=4, col="red")
yellows<-geom_sf(data=Cpt, size=4, col="yellow")
g+blues+reds+yellows
```



In order to apply duncan() to a sf we need to drop the geometry. global.duncan is simply that wrapper and is stored in "R/global.duncan.R"

```
global.duncan<-function(sf,PopA,PopB){
  df<-sf::st_drop_geometry(sf) #
  duncan(df,PopA,PopB)
}
```

Which, we now apply pairwise to our population groups:

```
list(global.duncan(sf,"A","B"),
global.duncan(sf,"A","C"),
global.duncan(sf,"B","C"))
```

```
## [[1]]
## [1] 0.3333333
##
## [[2]]
## [1] 0.5
##
## [[3]]
## [1] 0.6666667
```

Third example: making it local - local.duncan()

A local index relies on defining the neighbourhood of units where segregation is considered. Following spatial lag/dependence approaches we use a neighbourhood list (could be extended to a weight matrix but further complicates interpretation at this stage) based on contiguous neighbours.

Creating a nb list can be made using the sf route using sf::st_intersects or st_relate although it is quicker with spdep::poly2nb (see online discussion here: xxx). Not sure what the best route is but with the current examples we don't suffer any problem with sf. And we cannot see such local exploratory and visual approaches to be used to really massive data since the key is to observe and understand the local variations, an observation you can't really automate across many cases. Also segregation, especially localized, is mostly an intra-urban thing, not a regional cross-section one. Our real case example below has over 4000 units and runs very smoothly anyway

Also we use sf::st_interescts() here because each spatial unit under consideration must belong to its own neighbourhood (it is more a moving window approach than a spatial lag variable in that sense). For complex polygons, whether the rook or queen neighbourhood case is considered seems a rather marginal issue.

A contiguous neighbours list is thus computed as follows (without spdep):

```
#sgbp<-st_relate(sf, sf, pattern = "F***T****") #queen nb but would not include self
sgbp<-sf::st_intersects(sf,sf) #includes self

as.nb.sgbp <- function(x, ...) { #From Roger Bivand spdep vignette
  attrs <- attributes(x)
  x <- lapply(x, function(i) { if(length(i) == 0L) 0L else i } )
  attributes(x) <- attrs
  class(x) <- "nb"
  x
}
nb<-as.nb.sgbp(sgbp)
head(nb)
```

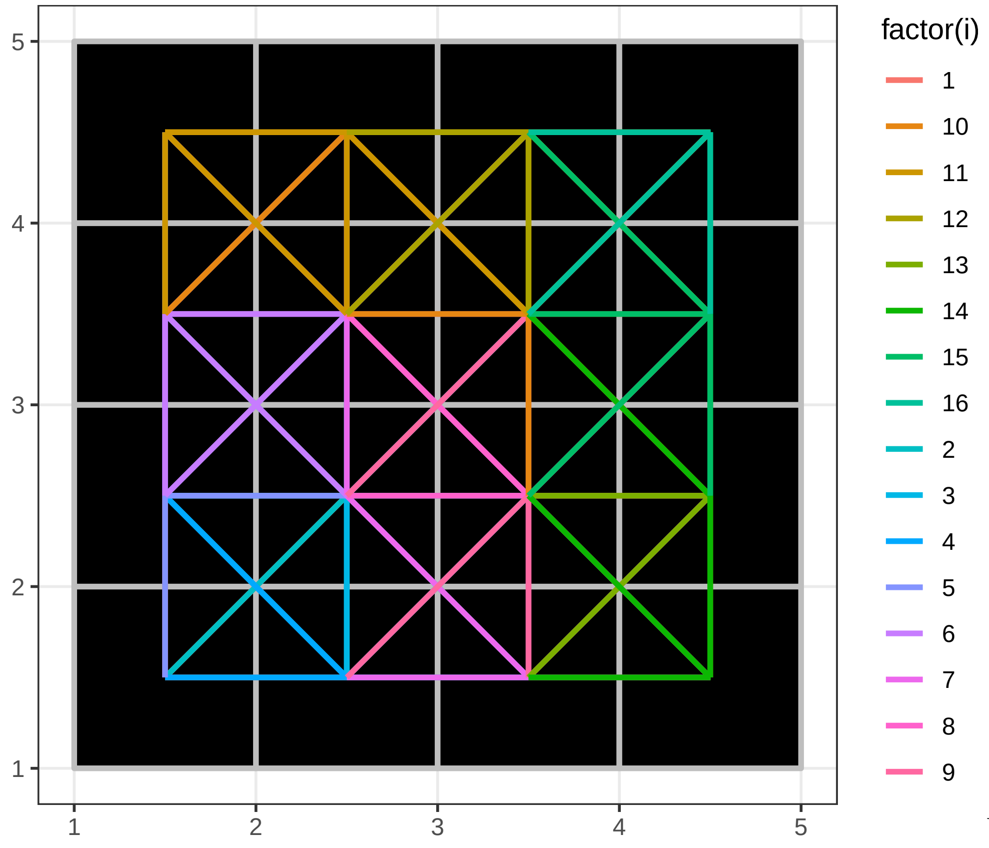
```
## [[1]]
## [1] 1 2 5 6
##
## [[2]]
## [1] 1 2 3 5 6 7
##
## [[3]]
## [1] 2 3 4 6 7 8
##
## [[4]]
## [1] 3 4 7 8
##
## [[5]]
## [1]  1  2  5  6  9 10
##
## [[6]]
## [1]  1  2  3  5  6  7  9 10 11
```

One can also map those neighbours' links. nb objects have a plot methods when coords are added. Here we first create a line sf from the neighbours list to view those segments. See "R/nbpoly2sflines.R"

```
nbpoly2sflines<-function(sf,nb){
  sf$nb.lst<-nb
  sf$i.sf<-rownames(sf)
  sf$X<-sf::st_coordinates(sf::st_centroid(sf))[,"X"]
  sf$Y<-sf::st_coordinates(sf::st_centroid(sf))[,"Y"]

  #nb list in long format: each link becoming a record
  #DOES NOT SEEM TO WORK IF EVERYONE IS EVERYONE's NEIGHBOUR. NOT SURE WHY YET.
  #Must be checked I expected it could be used to check global index is then found everywhere in this l
  long.nb<-data.frame(
    i=unlist(apply(sf,1,function(x){rep(x$i.sf,length(x$nb.lst))})),
    nb=unlist(sf$nb.lst))

  #merge xy of destinations (nb)
  DestinM<-merge(long.nb,sf::st_drop_geometry(sf[,c("i.sf","X","Y")]), by.x="nb", by.y="i.sf", all=TRUE
  names(DestinM)[c(3,4)]<-c("Xnb","Ynb")

  #merge xy of origins (i)
  OriginDestM<-merge(DestinM,sf::st_drop_geometry(sf[,c("i.sf","X","Y")]), by.x="i", by.y="i.sf", all=T
  names(OriginDestM)[c(5,6)]<-c("Xi","Yi")

  #Make a line geometry and sf
  long.nb$geom<-sprintf("LINESTRING(%s %s, %s %s)",
                        OriginDestM$Xi, OriginDestM$Yi, OriginDestM$Xnb, OriginDestM$Ynb)
  sf_nb_lines<-sf::st_as_sf(long.nb, wkt = "geom")

  return(sf_nb_lines)
}
```

Which, we now apply and map ggplot style (with colors per origin point)

```
nblines<-nbpoly2sflines(sf,nb)

library(ggplot2)
g2<-g+geom_sf(data=nblines, aes(col=factor(i)), linewidth=1)
g2
```

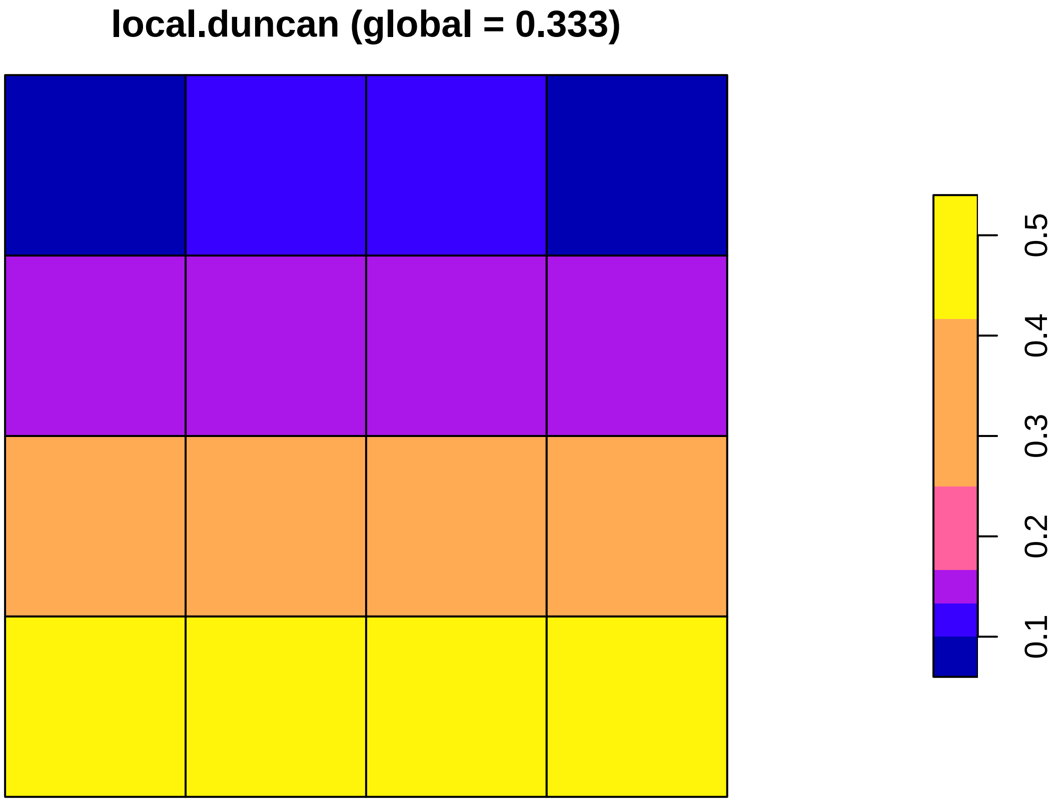


We now have the necessary ingredients fo a local Duncan. The function takes an sf, the population counts as the global one but also a corresponding nb list. See "R/local.duncan.R". Additional argument keep.sf is made to facilitate mapping and further plotting (also includes a sf based plot). Otherwise the local.moran values for each unit is returned.

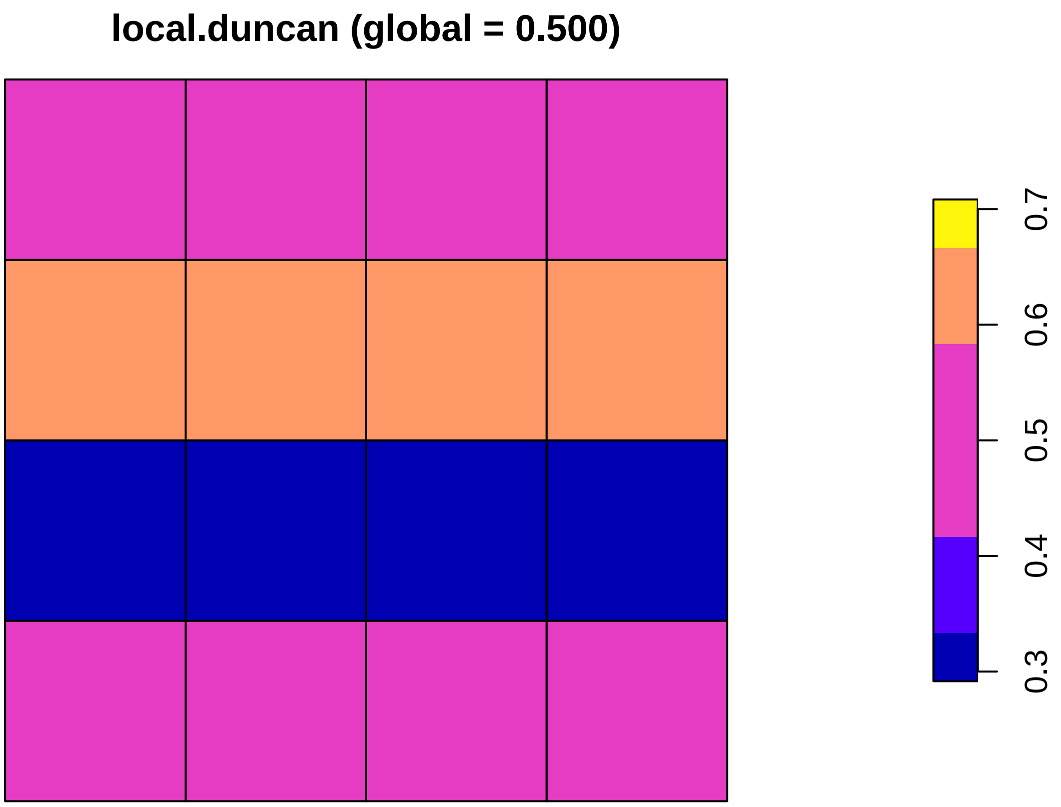
```r
local.duncan<-function(sf,PopA,PopB,nb,keep.sf=TRUE){
  df<-sf::st_drop_geometry(sf)
  global.duncan<-duncan(df,PopA,PopB)

  localdf<-lapply(nb,function(x){df[x,c(PopA,PopB)]})
  ld<-lapply(localdf,function(x){duncan(x,PopA,PopB)})
  local.duncan=unlist(ld)
  if(keep.sf==FALSE){
    out<-data.frame(local.duncan=local.duncan)
  } else {
    sf$local.duncan<-local.duncan
    out<-sf
    plot(sf["local.duncan"], main=paste0("local.duncan (global = ",
                                     sprintf("%.3f",global.duncan),")"))
  }
  return(out)
}
```

It is now applied:

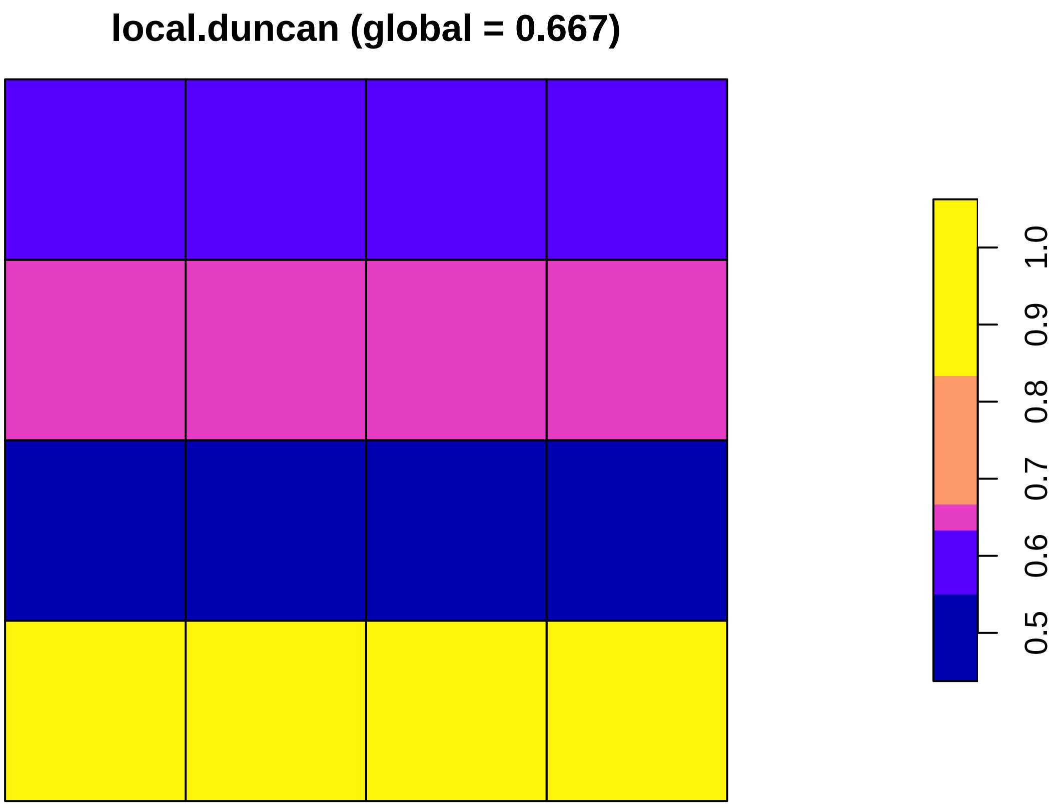
```r
Duncan_nbAB<-local.duncan(sf,"A","B",nb)
```



```r
Duncan_nbAC<-local.duncan(sf,"A","C",nb)
```

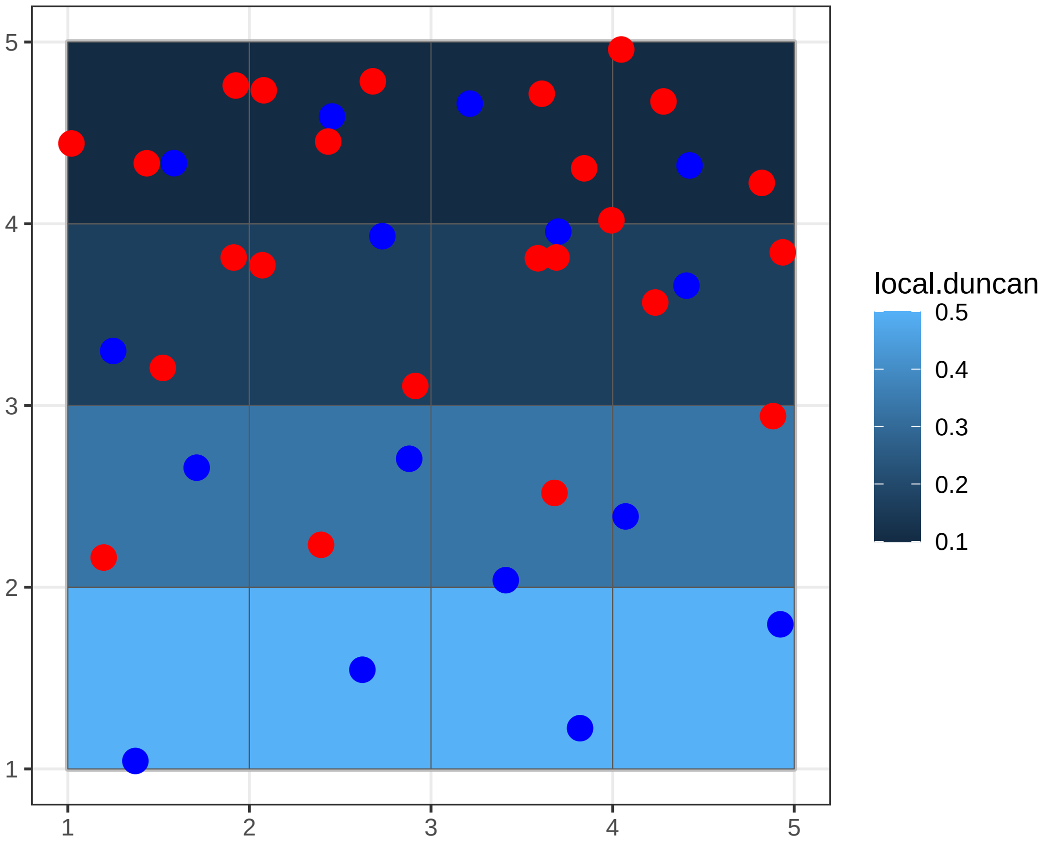


```r
Duncan_nbBC<-local.duncan(sf,"B","C",nb)
```



Similarly with ggplot for A and B

```r
g3<-g+geom_sf(data=Duncan_nbAB, aes(fill=local.duncan))
g3+blues+reds
```

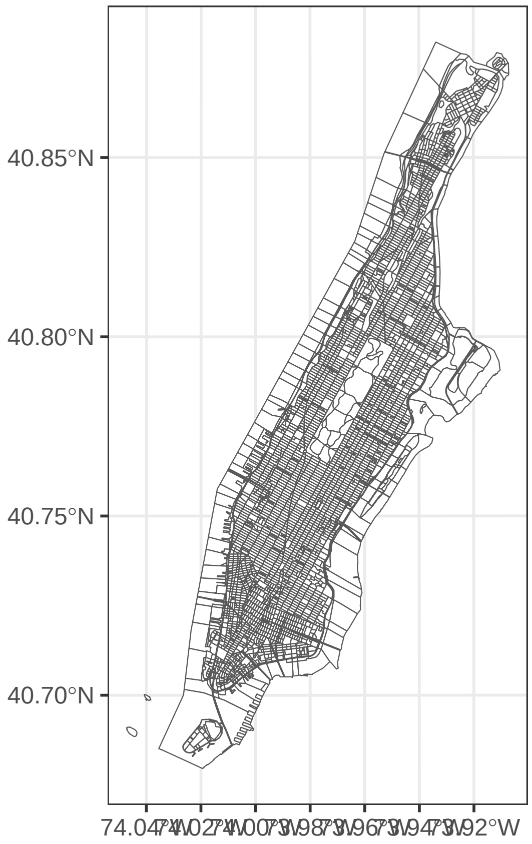


Real case - NY- Manhattan

We now introduce a real case study through which a related scatterplot function is also added, to support interpretation and understanding the behaviour of the metric

The data is about ethnicity per block in Manhattan. We brought the blocks spatial units beforehand from the package Tigris (see commented script), change into a local projection, and re-read it from local file.

```r
#NYMAN<-tigris::blocks(state="NY",county = "061")
#MAN2263<-sf::st_transform(NYMAN,crs=2263)
#saveRDS(MAN2263, "data/MAN2263.rds")
MAN2263<-readRDS("../data/MAN2263.rds") #re-read from local

gM<-ggplot()+geom_sf(data=MAN2263, fill="white")+theme_bw()
gM
```



The census data is obtained from https://s-media.nyc.gov/agencies/dcp/assets/files/excel/data-tools/census/census2020/nyc-decennialcensusdata_2010_2020_census-blocks.xls. It is a 100Mb file for the state with subtotals and many demographics. We externally subsetted to Manhattan and to the ethnic data only. From the metadata in the spreadsheet we have the variable names (P suffixes added for percentages within a block): Total population Pop2, Hispanic Hsp1, White non-Hispanic WNH, Black non-Hispanic BNH, Asian non-Hispanic ANH, Other race non-Hispanic ONH, non-Hispanic of two or more races TwoPlNH.

```
ETHN<-read.csv2("../data/Manhattan_ethn.csv")[-1,]
names(ETHN)
```

```
##  [1] "FIPSCode" "BoroCode" "GeogName" "GeoID"    "Pop2"     "Pop2P"
##  [7] "Hsp1"     "Hsp1P"    "WNH"      "WNHP"     "BNH"      "BNHP"
## [13] "ANH"      "ANHP"     "ONH"      "ONHP"     "TwoPlNH"  "TwoPlNHP"
```

We make the id's from within the sf comparable to those in the census and merge

```
MAN2263$GeoID<-substr(MAN2263$GEOID20, 5,16)
all(MAN2263$GeoID %in% ETHN$GeoID) #check match
```

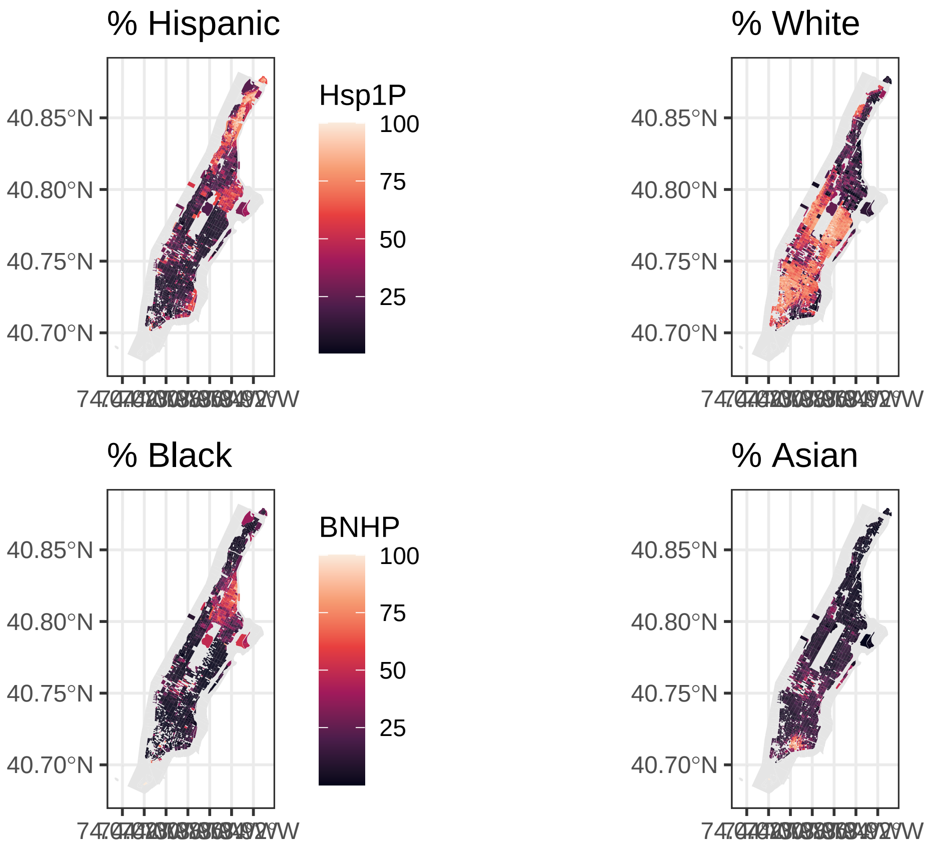
```
## [1] TRUE
```

```
MAN<-merge(MAN2263,ETHN, by="GeoID")
```

And map the percentage of some groups within each block:

```
gM_Hsp1P<-ggplot()+
  geom_sf(data=MAN, aes(fill=Hsp1P), col="transparent")+
  theme_bw()+labs(title="% Hispanic")+
  scale_fill_viridis_c(option = "rocket", na.value="grey90")
```

```
gM_WNHP<-ggplot()+
  geom_sf(data=MAN, aes(fill=WNHP), col="transparent")+
  theme_bw()+labs(title="% White")+
  scale_fill_viridis_c(option = "rocket", na.value="grey90")

gM_BNHP<-ggplot()+
  geom_sf(data=MAN, aes(fill=BNHP), col="transparent")+
  theme_bw()+labs(title="% Black")+
  scale_fill_viridis_c(option = "rocket", na.value="grey90")

gM_ANHP<-ggplot()+
  geom_sf(data=MAN, aes(fill=ANHP), col="transparent")+
  theme_bw()+labs(title="% Asian")+
  scale_fill_viridis_c(option = "rocket", na.value="grey90")

gridExtra::grid.arrange(gM_Hsp1P, gM_WNHP, gM_BNHP, gM_ANHP,nrow = 2)
```

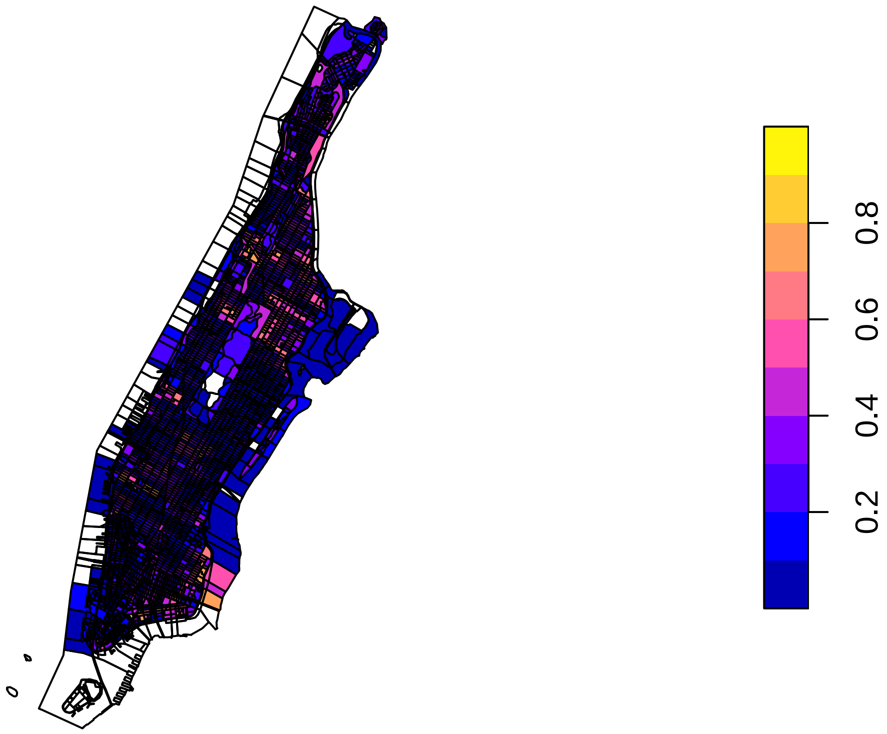


We can now compute the (global) Duncan index for some group pairs, then build a nb list and compute respective local Duncan's.

```
GDuncan_MAN_HW<-global.duncan(MAN,"Hsp1","WNH")
GDuncan_MAN_BW<-global.duncan(MAN,"BNH","WNH")
GDuncan_MAN_AW<-global.duncan(MAN,"ANH","WNH")
```

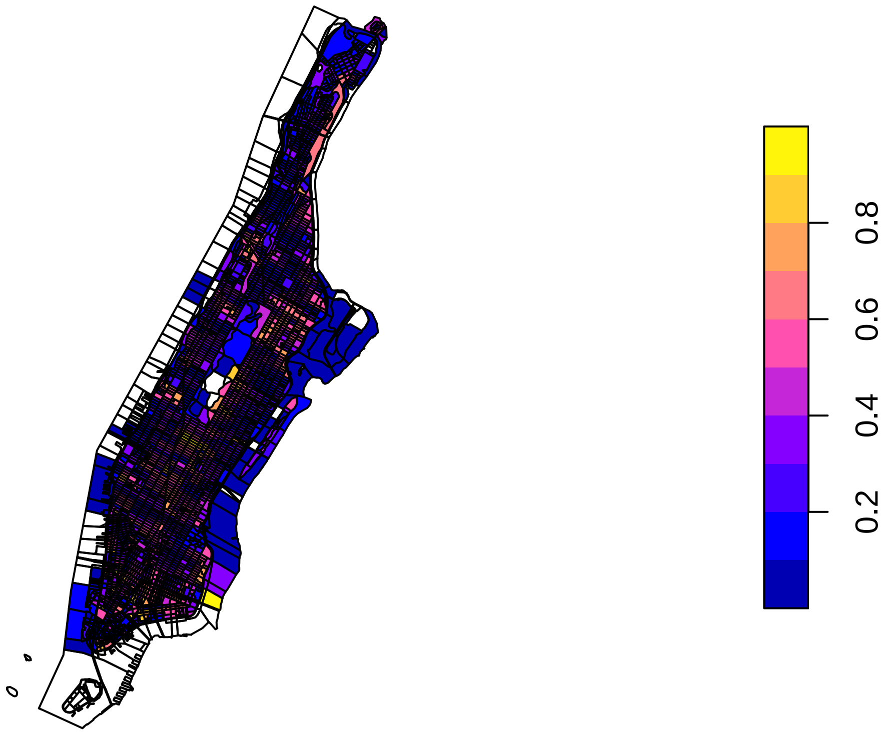
```
sgbpMAN<-sf::st_intersects(MAN, MAN)
nbMAN<-as.nb.sgbp(sgbpMAN)

Duncan_MAN_HW<-local.duncan(MAN,"Hsp1","WNH",nbMAN)
```

local.duncan (global = 0.621)



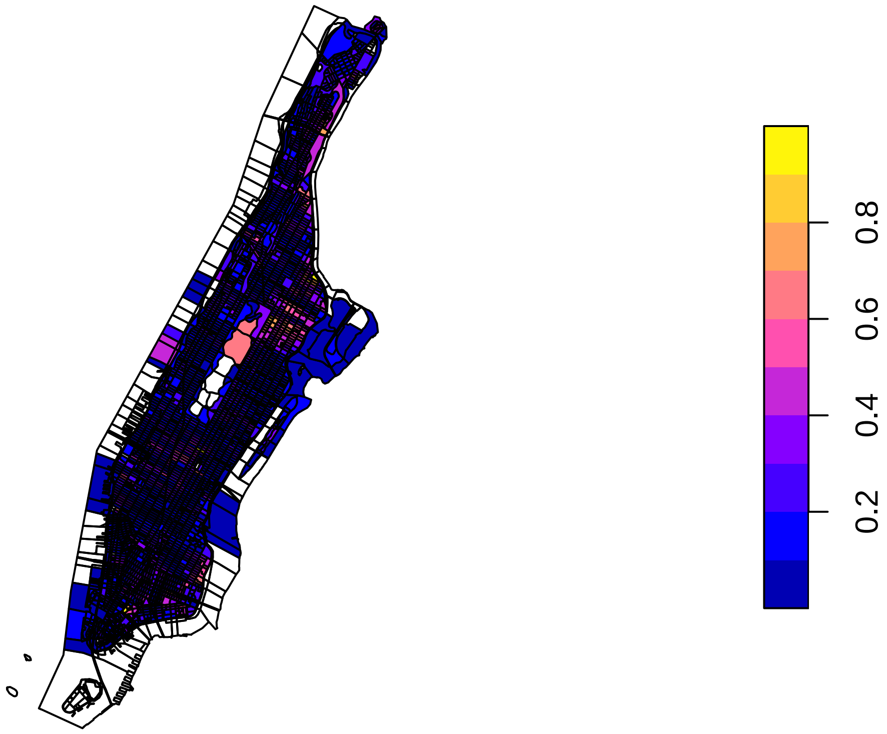
```
Duncan_MAN_BW<-local.duncan(MAN,"BNH","WNH",nbMAN)
```

local.duncan (global = 0.708)



```
Duncan_MAN_AW<-local.duncan(MAN,"ANH","WNH",nbMAN)
```

local.duncan (global = 0.375)



Finally, in an attempt to relate empirically the global index with local duncan values and with observed population shares, we devise a sort of duncan.scatterplot (kind of inspired by a moran.scatterplot).

The x-axis is the calculated share of group A in A+B for each spatial unit. The y-axis is the local duncan index for each spatial unit. A horizontal line is added to show the global duncan and a vertical line for the overall share of A in A+B (over the entire area)

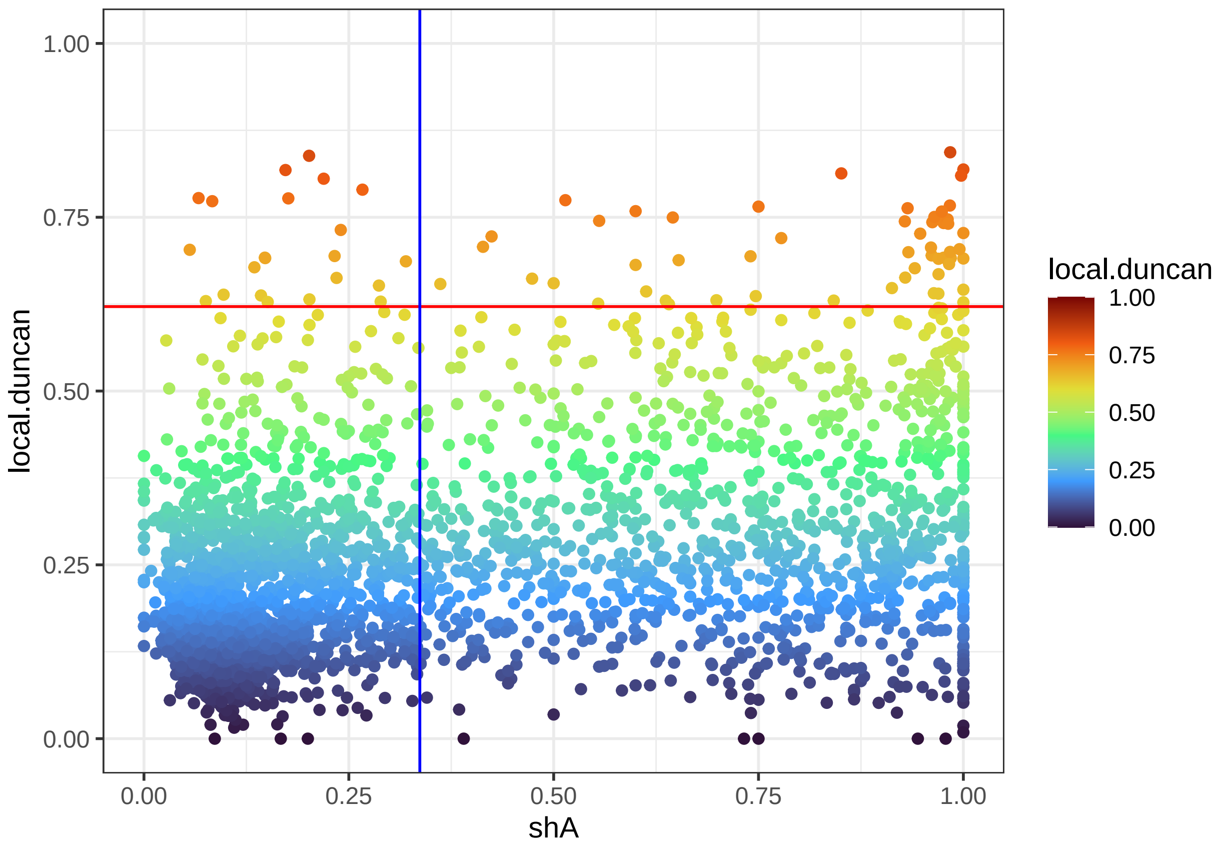
See “R/duncan.plot.R”. It is so far made in a ggplot manner but a base one would be wiser if the whole is turned into a package.

```
duncan.plot<-function(local.duncan.sf,PopA,PopB){
  df<-sf::st_drop_geometry(local.duncan.sf)
  A<-df[,PopA]
  B<-df[,PopB]
  df$shA<-A/(A+B)
  ggplot()+theme_bw()+
    geom_point(data=df, aes(x=shA, y=local.duncan, col=local.duncan))+
    scale_color_viridis_c(option = "turbo")+
    geom_hline(yintercept=global.duncan(local.duncan.sf,PopA,PopB), col="red")+
    geom_vline(xintercept=sum(A)/(sum(A)+sum(B)), col="blue")+
    coord_cartesian(xlim=c(0,1), ylim=c(0,1))
}
```

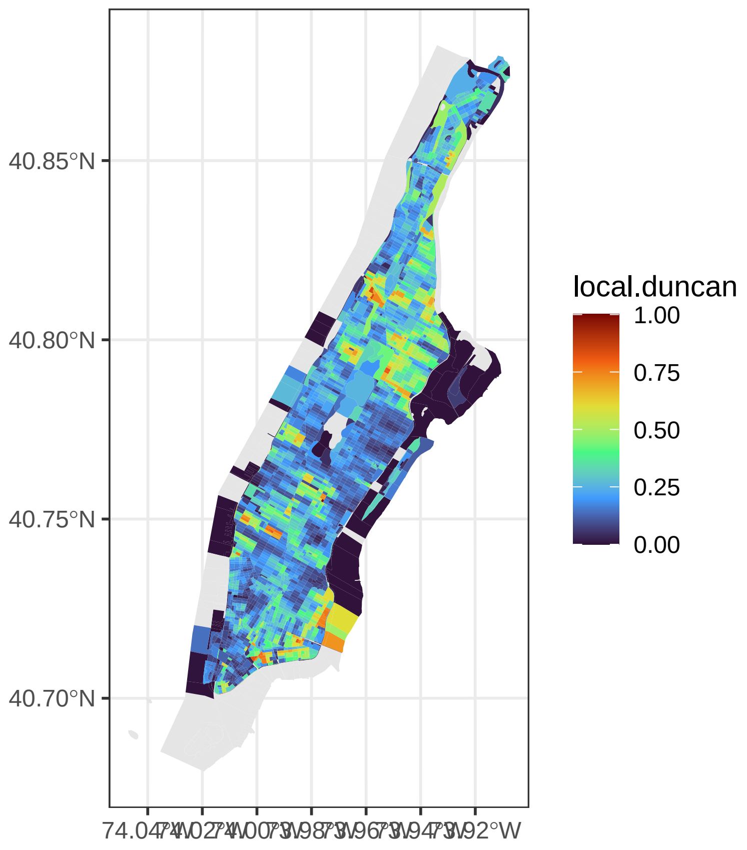
We apply the plot to the outcome of the local.duncan (with keep.sf on so a data.frame containing the original variables (for the x-axis) plus the local duncan (for the y-axis) is made).

The plots are made with colours following x (although it is bad practice to repeat information) so it can be associated with a map (ggplot way below)

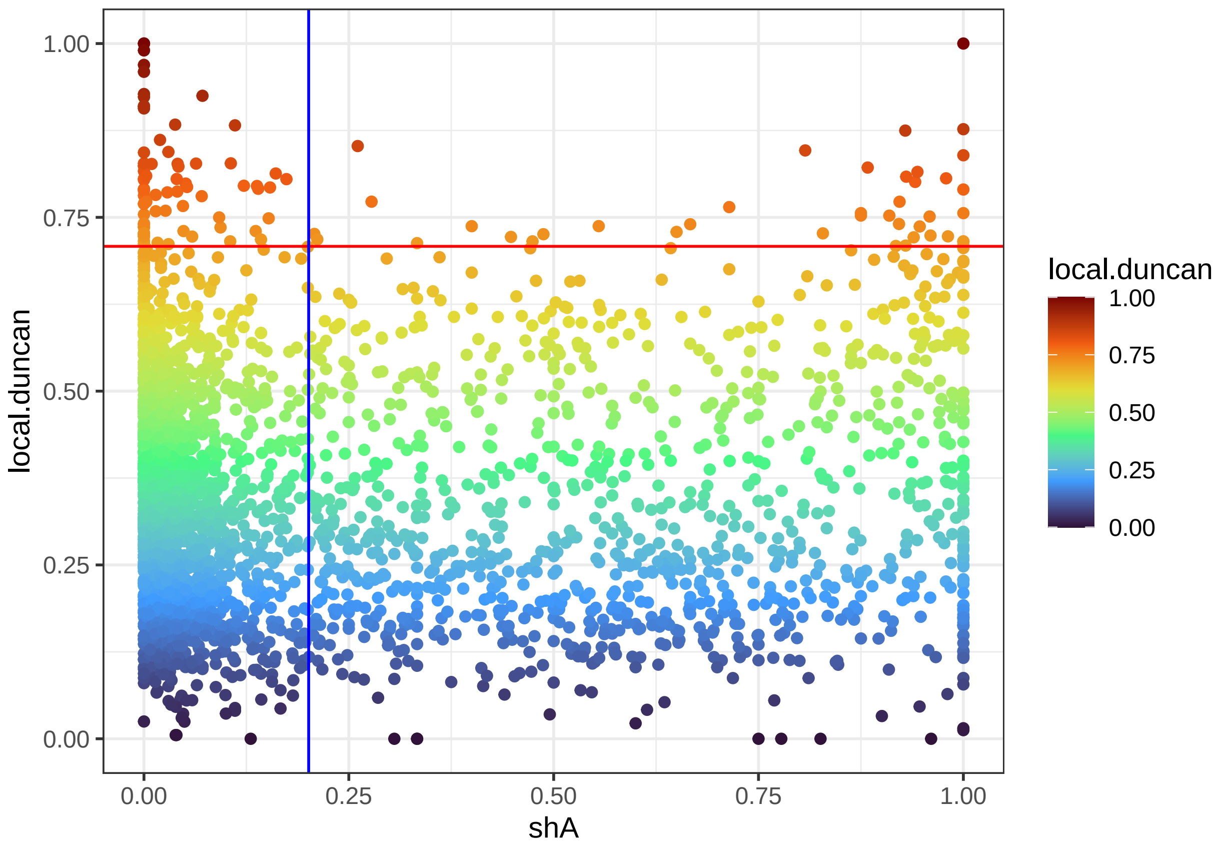
```
dpHW<-duncan.plot(Duncan_MAN_HW,"Hsp1","WNH")
dpHW
```



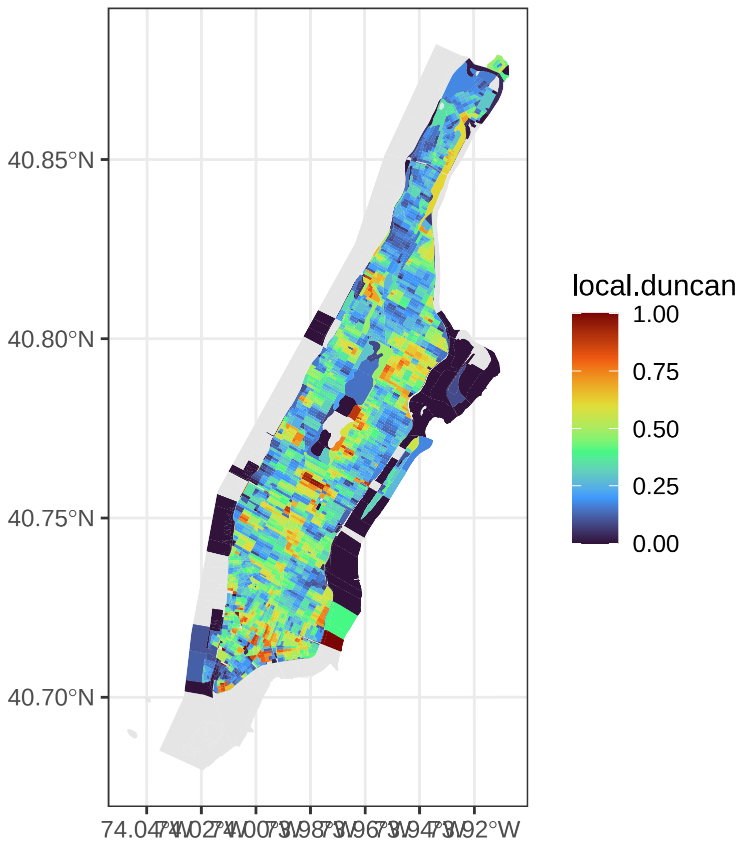
```
gM_HW<-ggplot()+
  geom_sf(data=Duncan_MAN_HW, aes(fill=local.duncan), col=NA)+
  theme_bw()+ scale_fill_viridis_c(option = "turbo", na.value="grey90")
gM_HW
```



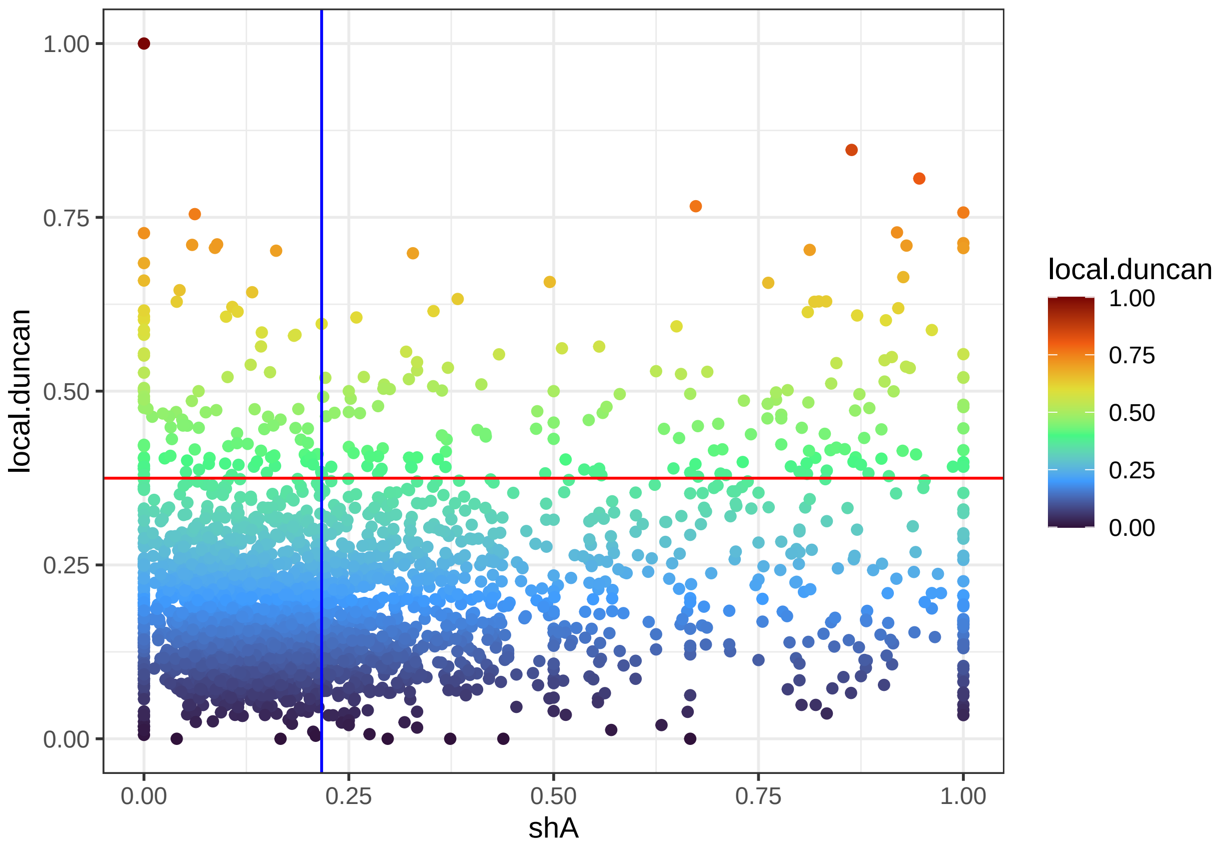
```r
dpBW<-duncan.plot(Duncan_MAN_BW,"BNH","WNH")
dpBW
```



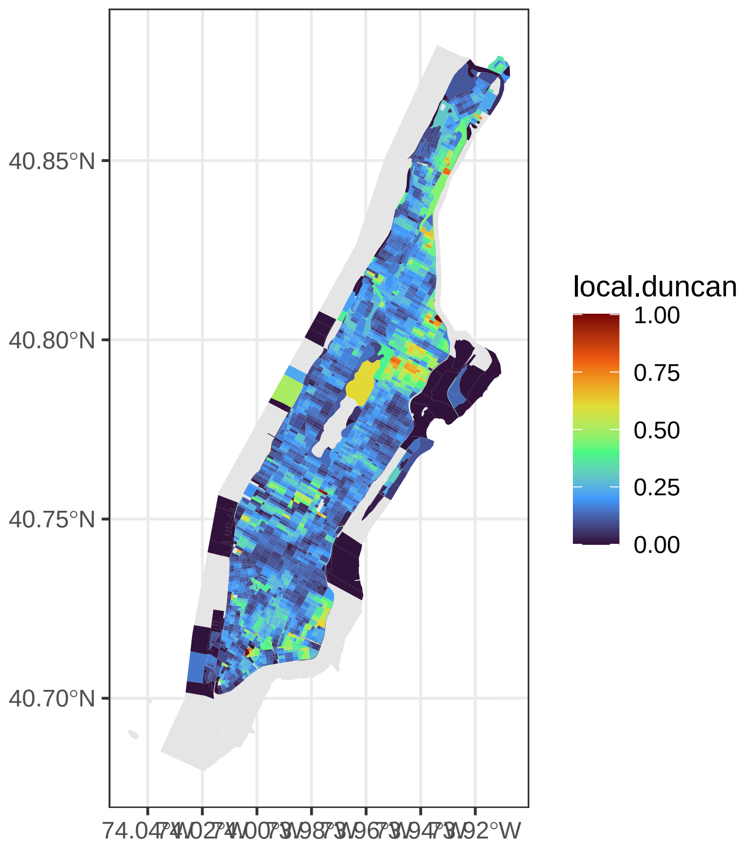
```
gM_BW<-ggplot()+
  geom_sf(data=Duncan_MAN_BW, aes(fill=local.duncan), col=NA)+
  theme_bw()+ scale_fill_viridis_c(option = "turbo", na.value="grey90")
gM_BW
```



```
dpAW<-duncan.plot(Duncan_MAN_AW,"ANH","WNH")
dpAW
```



```r
gM_AW<-ggplot()+
  geom_sf(data=Duncan_MAN_AW, aes(fill=local.duncan), col=NA)+
  theme_bw()+ scale_fill_viridis_c(option = "turbo", na.value="grey90")
gM_AW
```



From the plot one can see that when you pull apart horizontally the points to the left and to the right, there is more global segregation, i.e. the red bar is raised. Generally, because there is often some form of positive spatial autocorrelation it then means than on the local.moran axis most points are then below the bar, i.e below the general level. If daily routines happen in the given neighbourhood it is true that although a city can be very segregated, it may appear very homogeneous on the local level. Those points with higher local.moran happen to be at the connection between communities and the value somehow indicate how much reshuffling would be needed for a transition to be smoother.

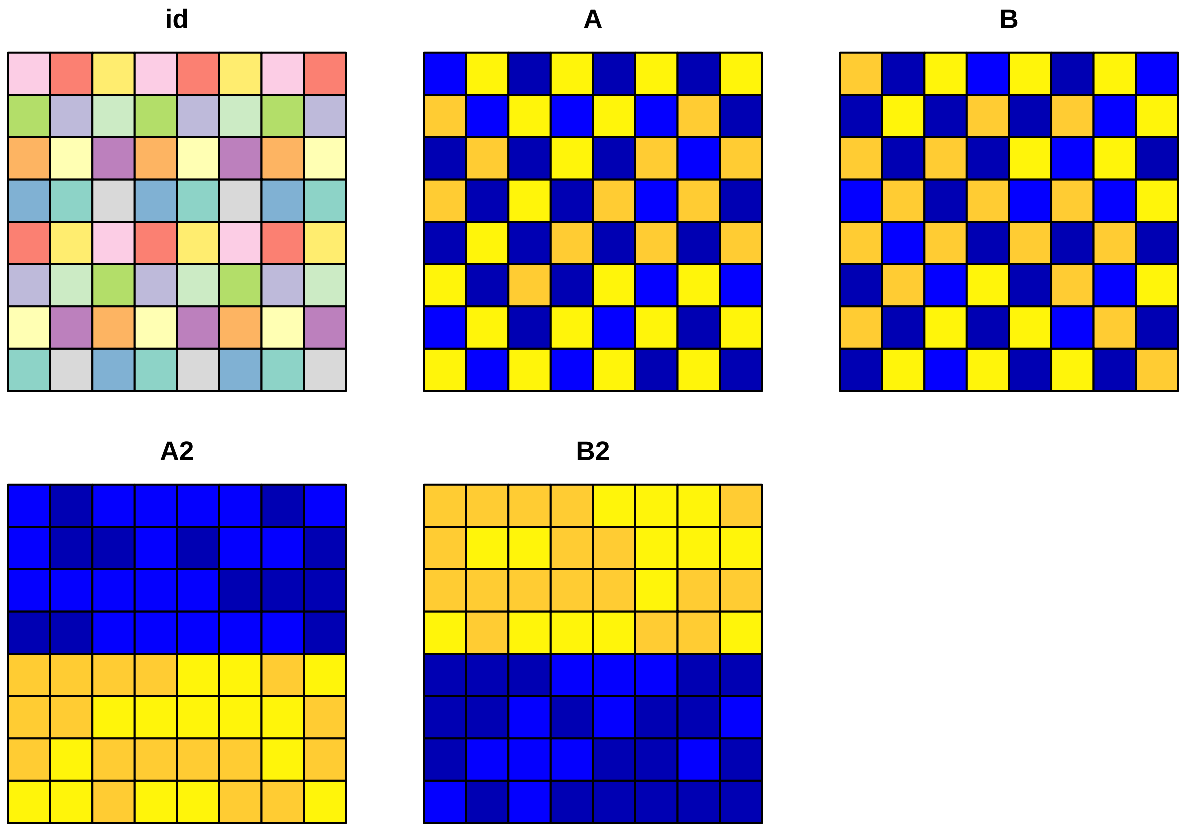
This calls for examining a case where the global duncan is the same but the local.ones high (a checkerboard) or low (grouped). From this we see this is not very far from a LISA (not the clusters, the values) but with the Duncan way of interpreting. Further digestion needed here!

Chessboard vs Zone field example

Same quantity (with a little random effect) of the two populations and same global duncan.

```r
x1<-rep((1:8),each=8)
y1<-rep((1:8),8)
df<-data.frame(id=paste(x1,y1,sep="_"))
df$geom<-sprintf("POLYGON((%s %s, %s %s, %s %s, %s %s, %s %s))",
                 x1, y1, x1, y1+1, x1+1, y1+1, x1+1, y1, x1, y1)
sf<-sf::st_as_sf(df, wkt = "geom")
#Populate
H<-100 #increase to differentiate from other and random effect (sd=1)
L<-20
sf$A<-rep(c(H,L,H,L,H,L,H,L,L,H,L,H,L,H,L,H),4)+rnorm(n=64)
sf$B<-rep(c(L,H,L,H,L,H,L,H,H,L,H,L,H,L,H,L),4)+rnorm(n=64)
sf$A2<-rep(c(H,H,H,H,L,L,L,L),8)+rnorm(n=64)
```

```r
sf$B2<-rep(c(L,L,L,L,H,H,H,H),8)+rnorm(n=64)
#Plot
plot(sf)
```



Two patterns. Same global Duncan:

```r
global.duncan(sf,"A","B")
```

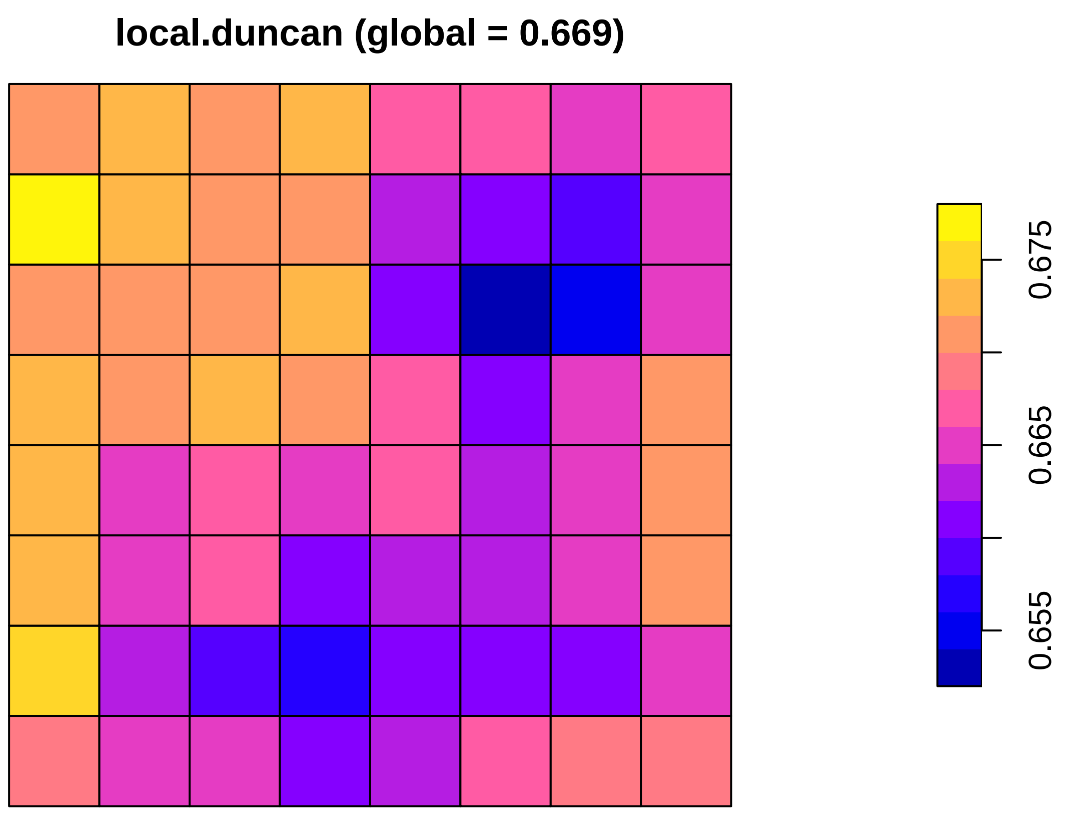
```
## [1] 0.6694577
```

```r
global.duncan(sf,"A2","B2")
```

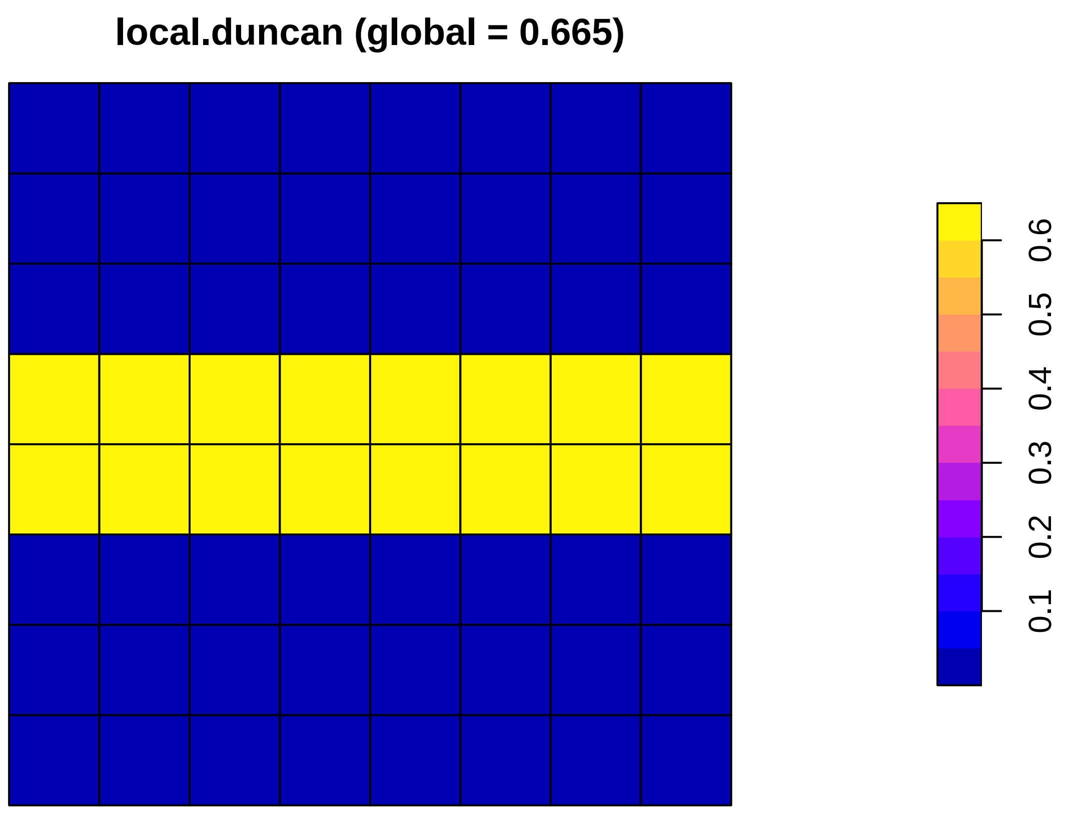
```
## [1] 0.6651384
```

Differing local.duncan (beyond border effect)

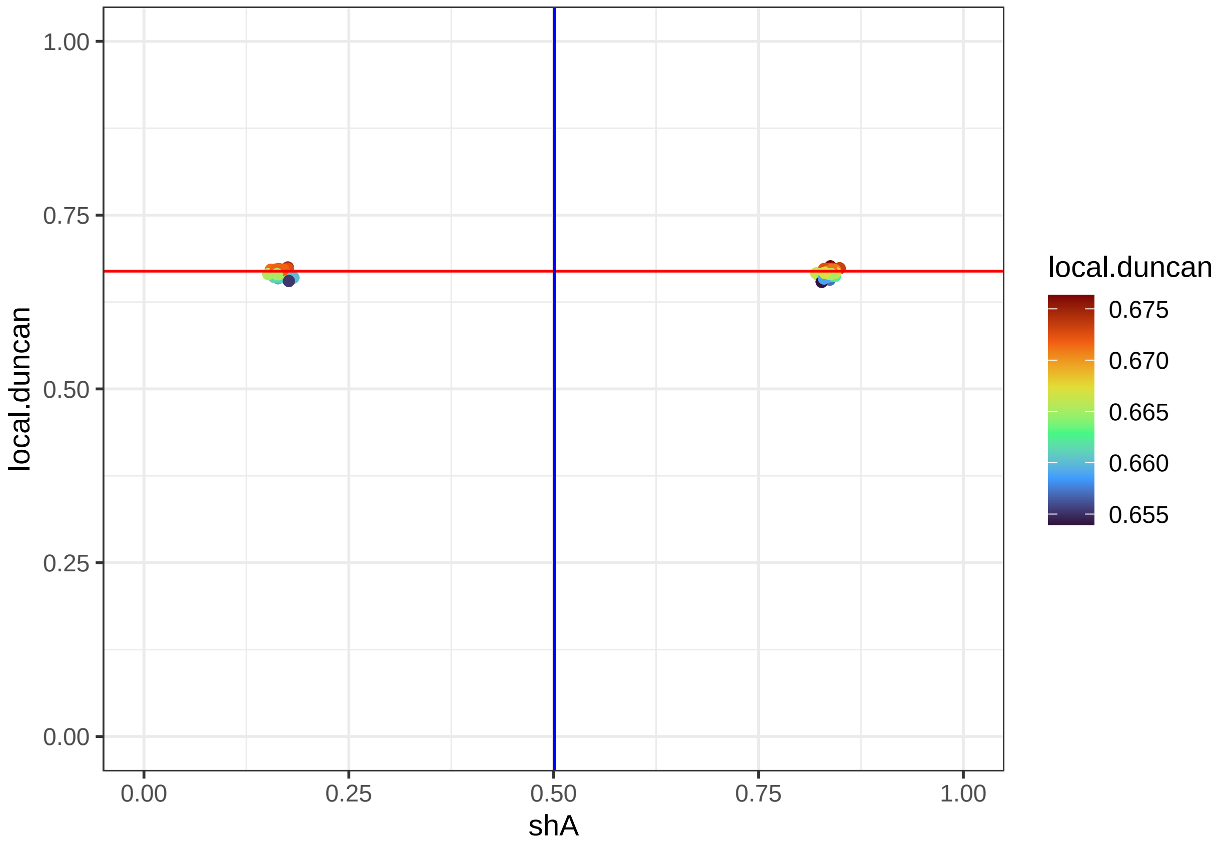
```r
sgbp<-sf::st_intersects(sf, sf)
nb<-as.nb.sgbp(sgbp)
ld<-local.duncan(sf,"A","B",nb)
```



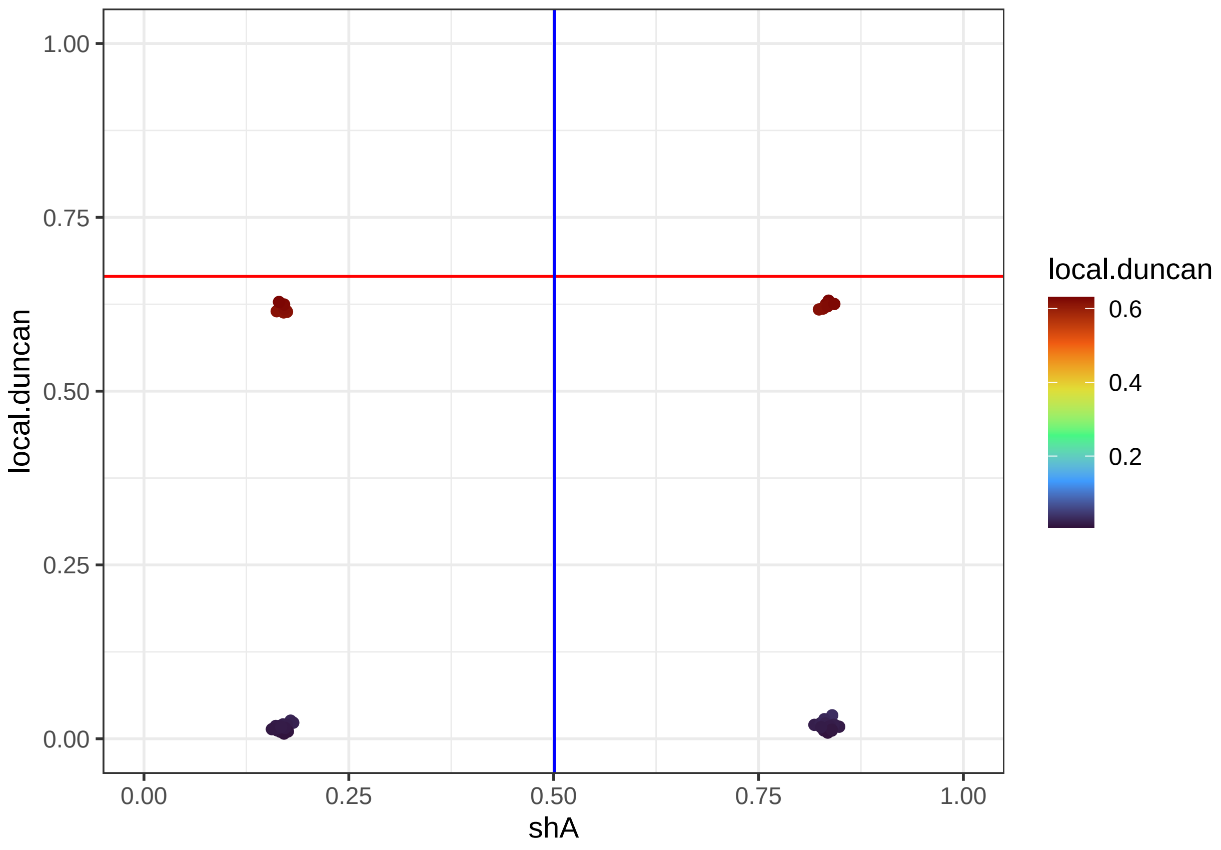
```
ld2<-local.duncan(sf,"A2","B2",nb)
```



```
dp<-duncan.plot(ld,"A","B")
dp2<-duncan.plot(ld2,"A2","B2")
dp
```



```
dp2
```


local.duncan
shA
1.00
0.75
0.50
0.25
0.00
0.6
0.4
0.2