Introduction to SSIMmap

The goal of the *SSIMmap* package extend the classical SSIM method for irregular lattice-based maps and raster images.The original SSIM develop to compare two image mimicking the human visual system and *SSIMmap* package applied this method to two types of maps (polygon and raster). A more generalizable SSIM method incorporates well-developed geographically weighted summary statisticsgeographically weighted summary statistics with an adaptive bandwidth kernel function for irregular lattice-based maps. This package includes four key functions: **ssim_bandwidth**, **ssim_constant**, **ssim_polygon**,**ssim_raster**. Users who want to compare two maps and quantify their similarities can utilize this package and visualize the results using other R packages(e.g.,*tmap* and *ggplot*).

Installation: SSIMmap

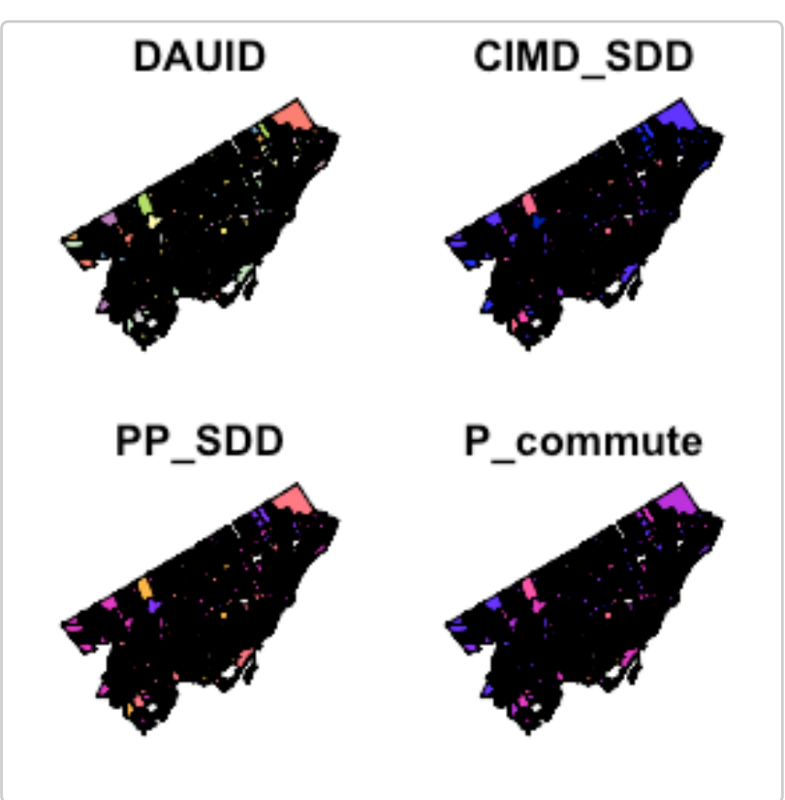
You can install the development version of *SSIMmap* from GitHub with the following commands:

```
devtools::install_github("Hailyee-Ha/SSIMmap")
library(SSIMmap)
```

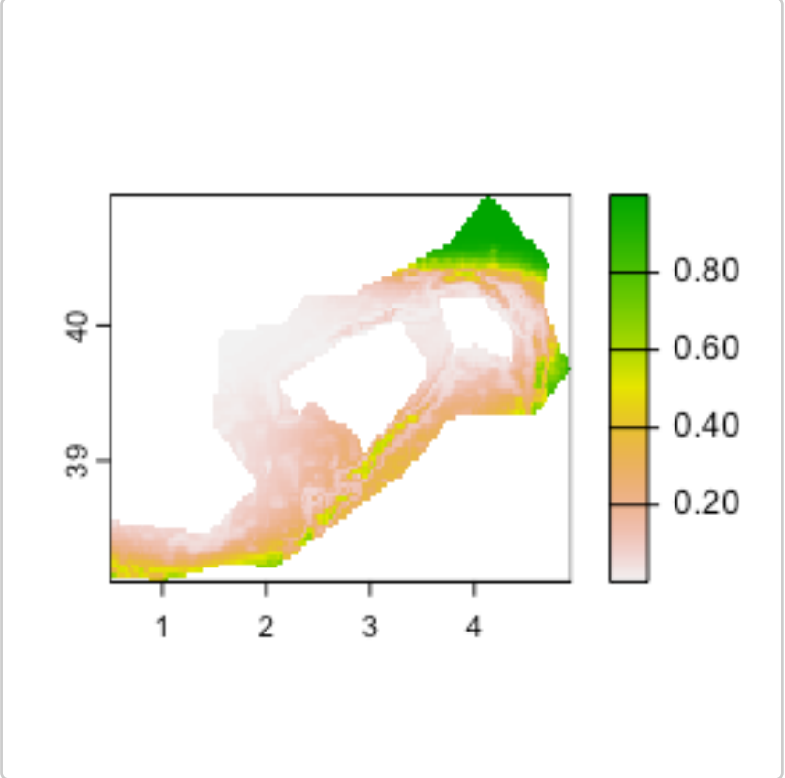
Data: polygon, groups2nm.tif, and single2nm.tif

The package has three example data: 1) polygon, 2) groups2nm, and 3) single2nm. First, **polygon** is an example data for *ssim_polygon*,*ssim_bandwidth*,and *ssim_constant*, which stands for Toronto, ON. This dataset contains neighborhood deprivation indices(the Canadian Index of Multiple Deprivation and Pampalon index) and the census variable(the percentage of households who commute within the census subdivision of residence) for 2016. Second and third data are image files for *ssim_raster* function. The image files are the location information of sperm whale as the presence of group or singletons.

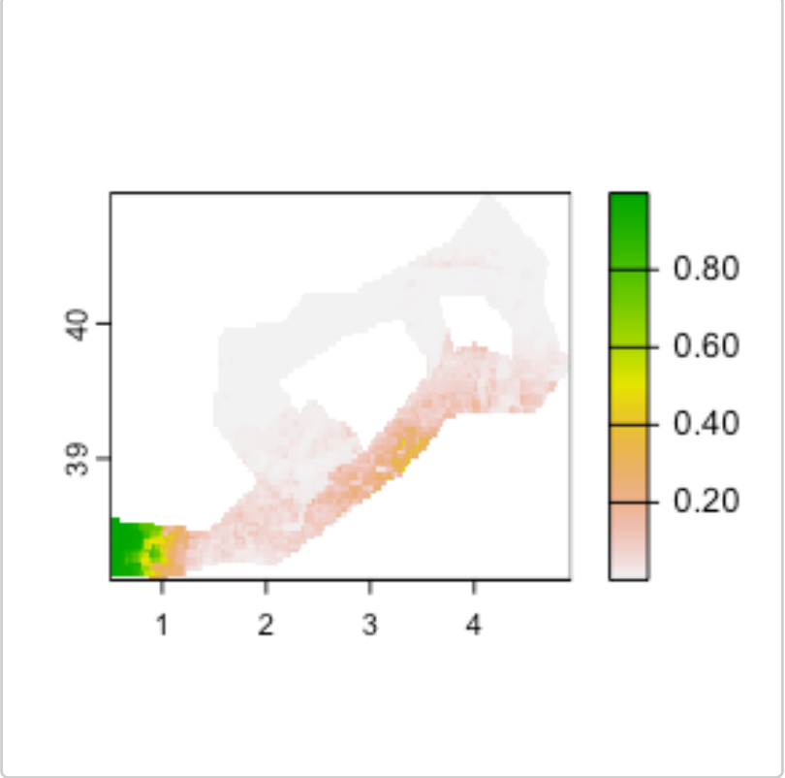
```
library(SSIMmap)
library(sf)
library(terra)
shape<- SSIMmap::polygon
plot(shape)
```



```
image_path1 <- system.file("data", "groups2nm.tif", package = "SSIMmap")
image_path2 <- system.file("data", "single2nm.tif", package = "SSIMmap")
img1<-terra::rast(image_path1)
img2<-terra::rast(image_path2)
plot(img1)
```



```
plot(img2)
```



Functions: ssim_bandwidth

The function ssim_bandwidth calculates the bandwidth size for the computation of the Structural Similarity Index (SSIM) on polygon maps. A user can determine whether the maps are standardized or not. The function provides two options for bandwidth size selection on the console window:

1. The square root of N.
2. The best trade-off between the bias and variance of the two maps.

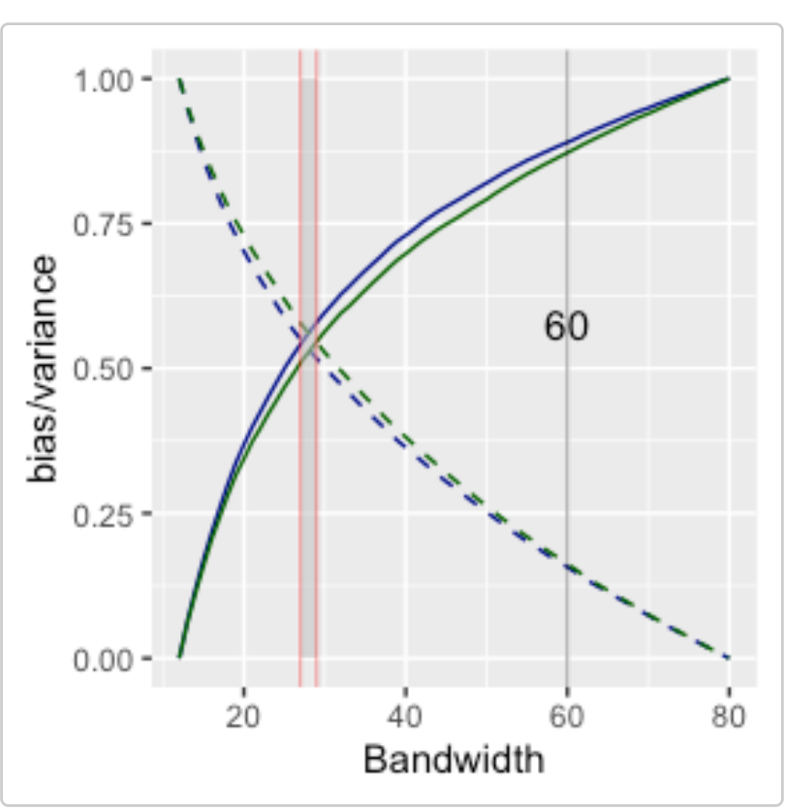
For the second method, the function defaults to the mid-point of the optimal trade-off between the bias and variance of map1 and map2. However, users can choose the upper (the larger one) or the lower (the smaller one) value between the two of the optimal trade-off between the bias and variance

The function takes as input a shape file in the *sf* class, which includes columns necessary for SSIM calculation. It outputs two possible selections for the bandwidth size based on the selected method. Additionally, the function generates a plot illustrating the relationship between bias, variance, and bandwidth size with a vertical line of the square root of N result.

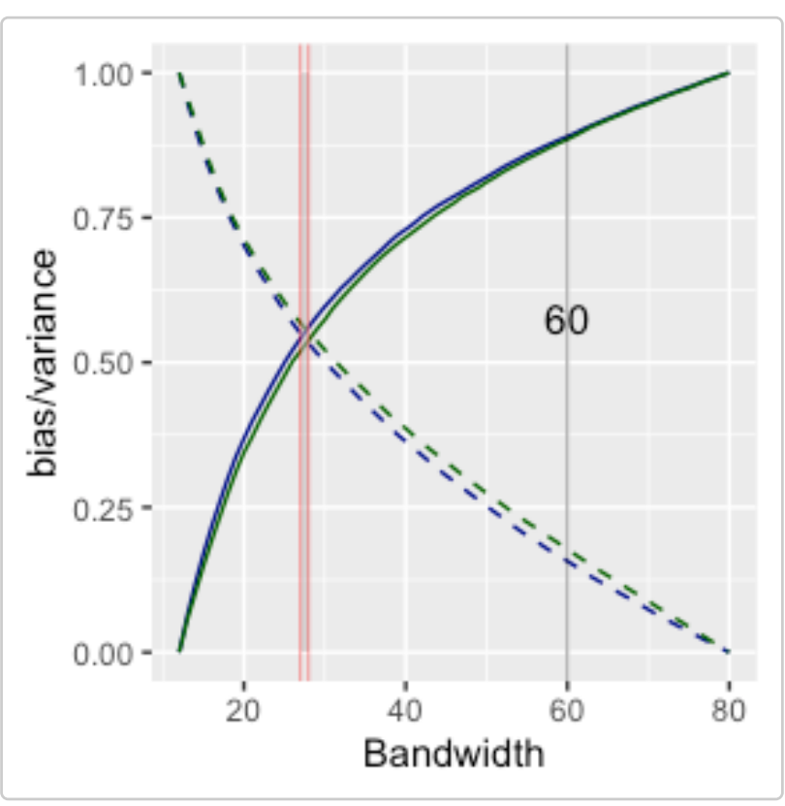
```
args(ssim_bandwidth)
#> function (shape, map1, map2, max_bandwidth = max_bandwidth, standarize = TRUE,
#>     option = "midpoint")
#> NULL
```

How to execute

```
ssim_bandwidth(shape,"CIMD_SDD","PP_SDD",max_bandwidth=80)
#> Square root N: 60 Bias-Variance Trade-off: 28
```



```
ssim_bandwidth(shape,"CIMD_SDD","P_commute",max_bandwidth=80)
#> Square root N: 60 Bias-Variance Trade-off: 28
```



Functions: ssim_constant

This function calculates constants(k1 and k2) for the computation of the SSIM on polygon maps based on the maximum value. It takes as input a shape file in *sf* class including columns for the SSIM calculation and returns the rescaled constants on the console window.

```
args(ssim_constant)
#> function (shape, map1, map2, standardize = TRUE)
#> NULL
```

How to execute

```
ssim_constant(shape,"CIMD_SDD","PP_SDD")
#> Rescaled K1: 0.00031 Rescaled K2: 0.00094
```

Functions: ssim_polygon

This function calculates the SSIM index for a given polygon. It takes a shape file in the *sf* class as input, which includes the necessary columns for SSIM calculation. The function then returns either the global SSIM values (when global = TRUE) or the local SSIM values for each given polygon (when global = FALSE). By default, the bandwidth size is determined by the square root of N. However, users have the flexibility to select their own bandwidth size or use the result from the *ssim_bandwidth* function.

```
args(ssim_polygon)
#> function (shape, map1, map2, standardize = TRUE, bandwidth = NULL,
#>     k1 = NULL, k2 = NULL, global = TRUE)
#> NULL
```

How to execute

```
ssim_polygon(shape,"CIMD_SDD","PP_SDD")
#>
#>
#> |Statistic |      SSIM|      SIM|      SIV|      SIP|
#> |:---------|---------:|---------:|---------:|---------:|
#> |Mean      | 0.5133388| 0.6705756| 0.9929326| 0.7724199|
#> |Min       | 0.2634429| 0.4865699| 0.8943711| 0.3995236|
#> |Max       | 0.6611332| 0.8835107| 1.0000000| 0.9081439|
#> |SD        | 0.0770358| 0.0776521| 0.0132320| 0.0897793|
ssim_polygon(shape,"CIMD_SDD","P_commute")
#>
#>
#> |Statistic |       SSIM|      SIM|      SIV|       SIP|
#> |:---------|----------:|---------:|---------:|----------:|
#> |Mean      | -0.0209320| 0.8463346| 0.9796295| -0.0279878|
#> |Min       | -0.3883281| 0.5824196| 0.9104065| -0.4738823|
#> |Max       |  0.4523199| 0.9973526| 1.0000000|  0.5901556|
#> |SD        |  0.1857759| 0.0723612| 0.0211420|  0.2263359|
df<-ssim_polygon(shape,"CIMD_SDD","PP_SDD",global = FALSE)
df_2<-ssim_polygon(shape,"CIMD_SDD","P_commute",global = FALSE)

head(df)
#> Simple feature collection with 6 features and 8 fields
#> Geometry type: MULTIPOLYGON
#> Dimension:     XY
#> Bounding box:  xmin: 1360108 ymin: 584839.9 xmax: 1360820 ymax: 585804.2
#> Projected CRS: Canada_Albers_Equal_Area_Conic
#>      DAUID CIMD_SDD      PP_SDD P_commute      SSIM       SIM       SIV
#> 1 35200002   -0.398 -0.02924313 0.2949438 0.3957402 0.5804081 0.9092756
#> 2 35200003   -0.784 -0.06272256 0.2537313 0.3966809 0.5810948 0.9095236
#> 3 35200004   -0.719 -0.06860705 0.3776042 0.3960310 0.5825955 0.9070322
#> 4 35200005   -0.737 -0.06804521 0.3676471 0.3968760 0.5834079 0.9069605
#> 5 35200006   -0.692 -0.05428556 0.2087287 0.3973070 0.5826379 0.9076319
#> 6 35200007   -0.769 -0.04231820 0.2909648 0.3984706 0.5837461 0.9082026
#>         SIP                       geometry
#> 1 0.7498617 MULTIPOLYGON (((1360419 585...
#> 2 0.7505513 MULTIPOLYGON (((1360570 585...
#> 3 0.7494442 MULTIPOLYGON (((1360409 585...
#> 4 0.7500569 MULTIPOLYGON (((1360346 585...
#> 5 0.7513075 MULTIPOLYGON (((1360731 585...
#> 6 0.7516049 MULTIPOLYGON (((1360574 584...
head(df_2)
#> Simple feature collection with 6 features and 8 fields
#> Geometry type: MULTIPOLYGON
#> Dimension:     XY
#> Bounding box:  xmin: 1360108 ymin: 584839.9 xmax: 1360820 ymax: 585804.2
#> Projected CRS: Canada_Albers_Equal_Area_Conic
#>      DAUID CIMD_SDD      PP_SDD P_commute       SSIM       SIM       SIV
#> 1 35200002   -0.398 -0.02924313 0.2949438 -0.1428618 0.6886711 0.9769972
#> 2 35200003   -0.784 -0.06272256 0.2537313 -0.1474403 0.6894965 0.9777682
#> 3 35200004   -0.719 -0.06860705 0.3776042 -0.1481428 0.6906690 0.9756653
#> 4 35200005   -0.737 -0.06804521 0.3676471 -0.1524195 0.6916361 0.9760587
#> 5 35200006   -0.692 -0.05428556 0.2087287 -0.1535657 0.6906311 0.9771665
#> 6 35200007   -0.769 -0.04231820 0.2909648 -0.1573527 0.6924337 0.9774029
#>          SIP                       geometry
#> 1 -0.2123298 MULTIPOLYGON (((1360419 585...
#> 2 -0.2186998 MULTIPOLYGON (((1360570 585...
#> 3 -0.2198415 MULTIPOLYGON (((1360409 585...
#> 4 -0.2257807 MULTIPOLYGON (((1360346 585...
#> 5 -0.2275514 MULTIPOLYGON (((1360731 585...
#> 6 -0.2324996 MULTIPOLYGON (((1360574 584...
```

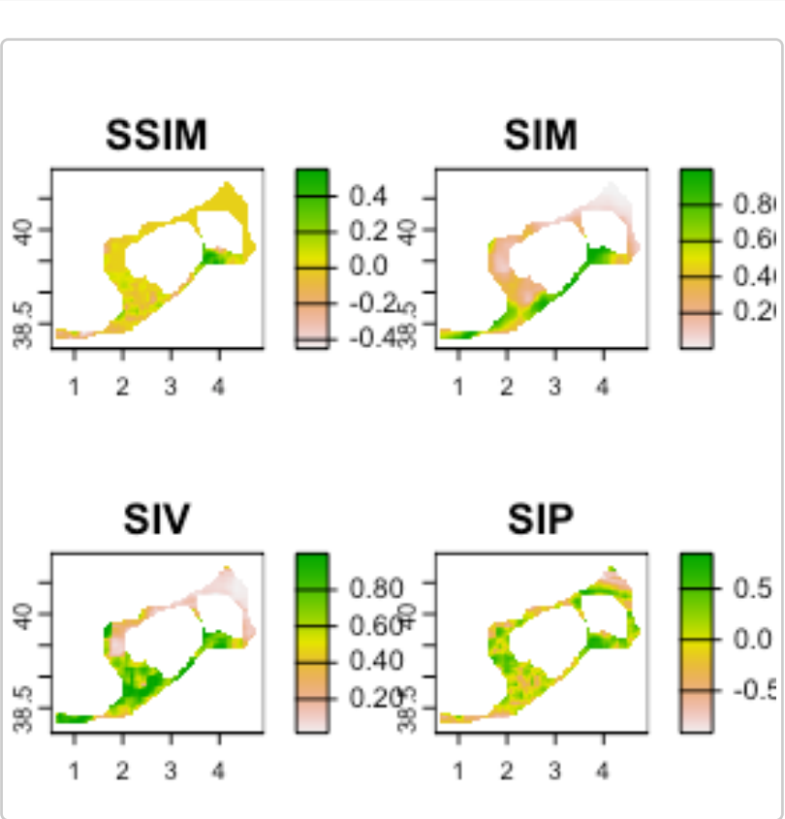
Functions: ssim_raster

This function calculates the SSIM index for raster images. It takes as input a image file importing from the *terra* and returns either the global SSIM values (global=TRUE) or the SSIM values for each given cell as the local SSIM (global=FALSE).Default of the window size is 3*3 and a user can use own window size.

```
args(ssim_raster)
#> function (img1, img2, global = TRUE, w = 3, k1 = NULL, k2 = NULL)
#> NULL
```

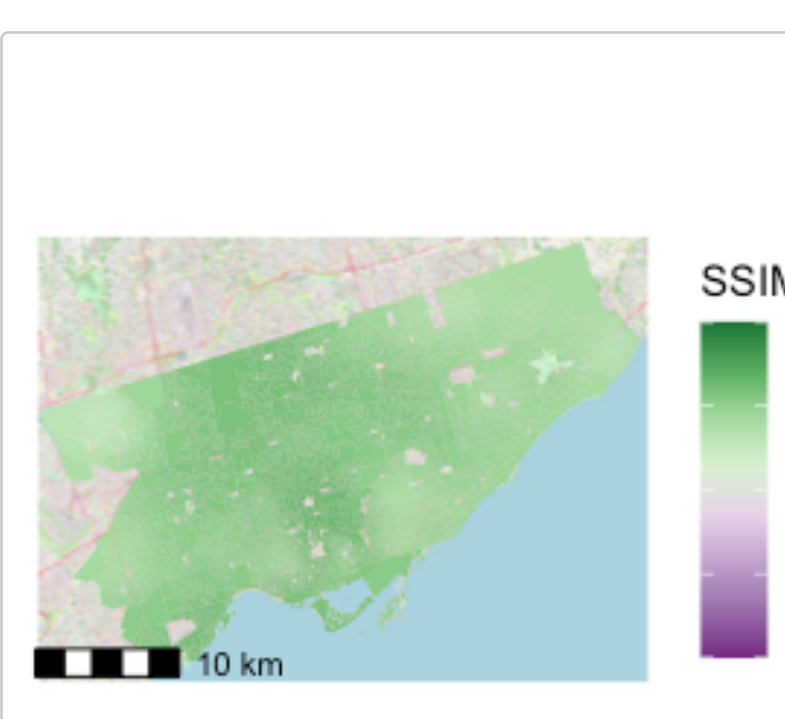
How to execute

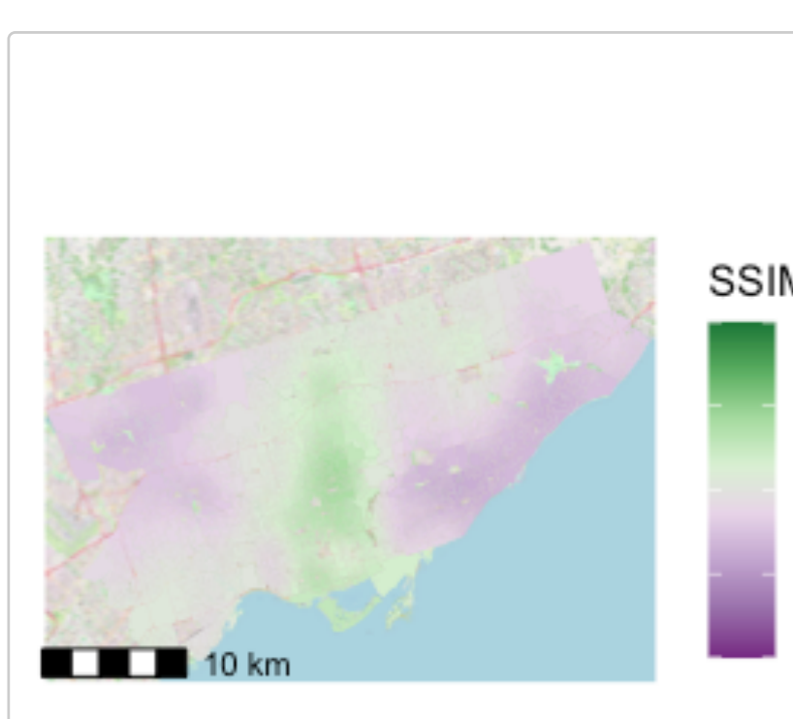
```
ssim_raster(img1,img2)
#> SSIM: 0.01788 SIM: 0.3739 SIV: 0.49631 SIP: 0.02351
result_raster<-ssim_raster(img1,img2,global = FALSE)
plot(result_raster)
```



Visualization of the local SSIM

```
library(maptiles)
library(tidyterra)
library(ggspatial)
library(ggplot2)
library(RColorBrewer)

#Transform shape(Toronto City) to a Mercator projection (EPSG:3857)
shape_NDI <- st_transform(shape, crs = 3857)
shape_NDI_valid <- st_make_valid(shape_NDI)
#Get the tiles for the background map
Toronto <- get_tiles(shape_NDI_valid, provider = "OpenStreetMap", zoom = 15)
#>
|---------|---------|---------|---------|
===========================================


#Define the palfunc function that creates a color palette to be used for the map visualization
 palfunc <- function (n, alpha = 1, begin = 1, end = 0, direction = 1)
  {
   #Create a 8-color palette from the RColorBrewer package called "PRGn (Purple to Green)"
    colors <- RColorBrewer::brewer.pal(8, "PRGn")
    if (direction < 0) colors <- rev(colors)
    colorRampPalette(colors, alpha = alpha)(n)
 }

#Visualize the SSIM result (the CIMD vs. Pampalon) on the map
ggplot() +
  geom_spatraster_rgb(data = Toronto) +
  geom_sf(data = df, aes(fill = SSIM), color = NA) +
  scale_fill_gradientn(colors = palfunc(8), limits = c(-1, 1)) +
  theme_void() +
  theme(
    axis.title.x = element_blank(),
    axis.title.y = element_blank(),
    axis.text.x = element_blank(),
    axis.text.y = element_blank(),
    axis.ticks = element_blank()
  )+ ggspatial::annotation_scale()

#Visualize the SSIM result (the CIMD vs. commuting pattern) on the map
ggplot() +
  geom_spatraster_rgb(data = Toronto) +
  geom_sf(data = df_2, aes(fill = SSIM), color = NA) +
  scale_fill_gradientn(colors = palfunc(8), limits = c(-1, 1)) +
  theme_void() +
  theme(
    axis.title.x = element_blank(),
    axis.title.y = element_blank(),
    axis.text.x = element_blank(),
    axis.text.y = element_blank(),
    axis.ticks = element_blank()
  )+ ggspatial::annotation_scale()
```





Reference

- Brunsdon, C., Fotheringham, A.S. and Charlton, M. (2002). Geographically weighted summary statistics—a framework for localised exploratory data analysis. *Computers, Environment and Urban Systems*, 26(6), pp.501-524. https://doi.org/10.1016/S0198-9715(01)00009-6
- Jones, E.L., Rendell, L., Pirotta, E. and Long, J.A. (2016). Novel application of a quantitative spatial comparison tool to species distribution data. *Ecological Indicators*, 70, pp.67-76. https://doi.org/10.1016/j.ecolind.2016.05.051
- Lu, B., Harris, P., Charlton, M. and Brunsdon, C. (2014). The GWmodel R package: further topics for exploring spatial heterogeneity using geographically weighted models. *Geo-spatial Information Science*, 17(2), pp.85-101. https://doi.org/10.1080/10095020.2014.917453
- Wang, Z., Bovik, A.C., Sheikh, H.R. and Simoncelli, E.P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4), pp.600-612. https://doi.org/10.1109/TIP.2003.819861