

Manual de uso de Makurhini

Oscar Godínez Gómez

2025-07-18

Contents

Makurhini	5
1 Instalación del paquete Makurhini	7
1.1 Instalación estandar	7
1.2 Instalar en Linux	8
2 Estadísticas de fragmentación del paisaje	11
2.1 Inputs	11
2.2 Funcion	14
2.3 Ejercicio 1	14
2.4 Ejercicio 2	26
2.5 Ejercicio 3	28
3 Índices de centralidad	31
3.1 Insumos y paquetes	31
3.2 MK_RMCentrality()	32
3.3 Ejemplo 1	37
3.4 Ejemplo 2	42

Makurhini



En este manual se documenta parte de las funciones del paquete Makurhini

Chapter 1

Instalación del paquete Makurhini

1.1 Instalación estandar

- Depende de R (> 4.0.0), igraph (>= 1.2.6)

```
install.packages("igraph")
```

- Se recomienda pre-instalar Rtools: <https://cran.r-project.org/bin/windows/Rtools/>
- Se recomienda pre-instalar los paquetes devtools y remotes

```
install.packages(c("devtools", "remotes"))
```

```
library(devtools)
library(remotes)
install_github("connectscape/Makurhini", dependencies = TRUE, upgrade = "never")
```

En caso de que no aparezca en la lista de paquetes, cierre la sesión de R y vuelva a abrirla.

Si se produce el siguiente error durante la instalación:

```
Using github PAT
from envvar GITHUB_PAT Error: Failed to install 'unknown package' from
GitHub: HTTP error 401. Bad credentials
```

Entonces intenta lo siguiente:

```
Sys.getenv("GITHUB_PAT")
Sys.unsetenv("GITHUB_PAT")
```

1.2 Instalar en Linux

Makurhini en Linux Para instalar Makurhini en linux considere los siguientes pasos:

Utilice la línea de comandos de Linux para instalar el paquete de la unidad:

```
sudo apt-get install -y libudunits2-dev
```

Utilice la línea de comandos de Linux para instalar gdal:

```
sudo apt install libgdal-dev
```

Utilice la línea de comandos de Linux para instalar libfontconfig y libharfbuzz:

```
sudo apt install libfontconfig1-dev
sudo apt install libharfbuzz-dev libfribidi-dev
```

Ahora puede instalar los paquetes devtools y remotes, y los paquetes terra, raster y sf directamente en su R o RStudio.

```
install.packages(c('remotes', 'devtools', 'terra', 'raster', 'sf'))
```

Utiliza la línea de comandos de Linux para instalar igraph:

```
sudo apt-get install libnlopt-dev
sudo apt-get install r-cran-igraph
```

Ahora puede instalar los paquetes gdistance, graph4lg y ggpubr directamente en su R o RStudio.

```
install.packages(c('gdistance', 'graph4lg', 'ggpubr'))
```

Ahora puedes instalar Makurhini directamente en tu R o RStudio.

```
library(devtools)
library(remote)
install_github("connectscape/Makurhini", dependencies = TRUE, upgrade = "never")
```

Tenga en cuenta que la instalación de Makurhini en Linux depende de su versión del sistema operativo y de que consiga instalar los paquetes de los que depende Makurhini.

Chapter 2

Estadísticas de fragmentación del paisaje

Exploraremos la función `MK_Fragmentation()` para caracterizar la composición y la configuración espacial de los parches en nuestro paisaje.

La función calcula **ocho métricas a nivel de nodo** (como el área de nodo, el porcentaje de borde y la dimensión fractal) y **13 estadísticas de fragmentación a nivel de paisaje** (como el número de nodos, el tamaño medio, la densidad de borde y el tamaño efectivo de la malla).

Las entradas incluyen un **objeto vectorial** que representa los nodos del paisaje y un parámetro que define la distancia o profundidad de la influencia del borde (es decir, la distancia al borde del nodo).

Los resultados consisten en un **archivo vectorial tipo poligonal** que contiene métricas a nivel de nodo y **una tabla** que detalla las métricas a nivel de paisaje.

2.1 Inputs

Usaremos la librerías sf, para trabajar con un shapefile que contiene 404 parches de vegetación con la menor alteración o signos de degradación en un estado de México llamado Michoacán. También cargaremos el paisaje de estudio donde se encuentran los parches y que fue delimitado usando el límite político del estado y la región fisiográfica denominada Eje Neovolcánico.

El shapefile lo pueden encontrar en la siguiente carpeta del drive:

<https://drive.google.com/drive/folders/1yJcxk2JsEfVjqBx7QM6SV5sIbq7YD1iX?usp=sharing>

12 CHAPTER 2. ESTADÍSTICAS DE FRAGMENTACIÓN DEL PAISAJE

```
#> Cargando paquete requerido: igraph
#>
#> Adjuntando el paquete: 'igraph'
#> The following objects are masked from 'package:stats':
#>
#>     decompose, spectrum
#> The following object is masked from 'package:base':
#>
#>     union
#> Cargando paquete requerido: cppRouting
#> Linking to GEOS 3.13.0, GDAL 3.10.1, PROJ 9.5.1;
#> sf_use_s2() is TRUE
#> [1] 404
```

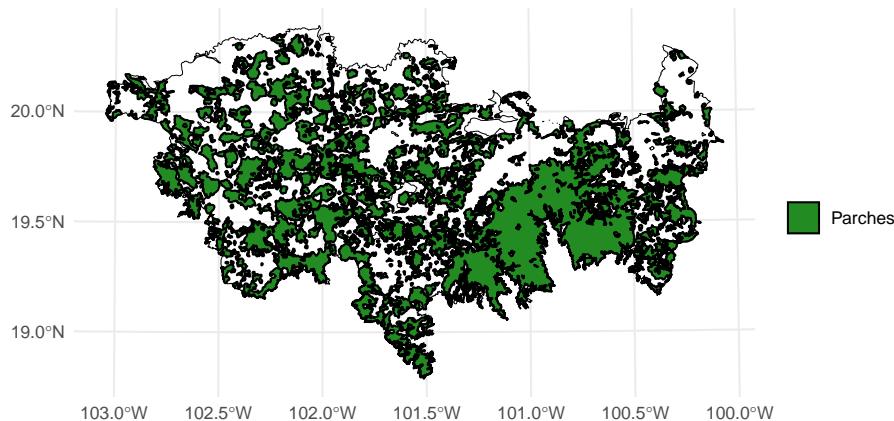
```
library(Makurhini)
library(sf)

habitat_nodes <- read_sf("C:/Users/habitat_nodes.shp")
nrow(habitat_nodes)
paisaje <- read_sf("C:/Users/paisaje.shp")
```

Para graficarlo necesitamos ggplot2, favor de instalarlo si no lo tiene entre sus paquetes.

```
install.packages("ggplot2"), dependencies = TRUE
install.packages("RColorBrewer"), dependencies = TRUE
```

```
library(ggplot2)
library(RColorBrewer)
ggplot() +
  geom_sf(data = paisaje, aes(color = "Study area"), fill = NA, color = "black") +
  geom_sf(data = habitat_nodes, aes(color = "Parches"), fill = "forestgreen", linewidth = 1) +
  scale_color_manual(name = "", values = "black") +
  theme_minimal() +
  theme(axis.title.x = element_blank(),
        axis.title.y = element_blank())
```

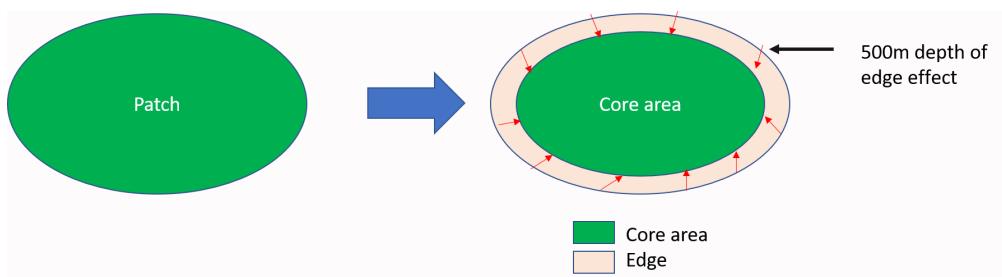


En caso de necesitar abrir otro vector (e.g., .shp, .gpkg) necesitan usar la fusión `read_sf()` del paquete `sf`, la función `shapefile()` del paquete `raster`, o la función `vect()` del paquete `terra`.

Para abrirlo solo necesitan colocar la dirección de su archivo, el nombre y la extensión, ejemplo:

- `vegetation_patches <- sf::read_sf("D:/Datos/vegetation_patches.shp")`
- `vegetation_patches <- raster::shapefile("D:/Datos/vegetation_patches.shp")`
- `vegetation_patches <- terra::vect("D:/Datos/vegetation_patches.shp")`

Para definir el borde usaremos una distancia de 500 m a partir del límite de los parches (Haddad et al. 2015).



2.2 Funcion

```
MK_Fragmentation(
  nodes = NULL,
  edge_distance = 500,
  min_node_area = 100,
  landscape_area = NULL,
  area_unit = "ha",
  perimeter_unit = "km",
  plot = FALSE,
  write = NULL
)
```

Los argumento de la función que usaremos son:

- *nodes* = objeto con los parches,
- *edge_distance* = profundidad del efecto de borde.
- *min_node_area* = Área mínima del nodo utilizada para calcular el número de nodos con un área menor a la proporcionada. Por defecto igual a 100 km² (Haddad et al. 2015).
- *landscape_area* = Área total del paisaje de estudio (opcional). Si se deja como NULL, se utilizará el área total de los nodos. La unidad de área debe ser igual a la seleccionada en el parámetro *area_unit*.
- *area_unit* = Puedes establecer una unidad de área (por ejemplo, “km²”, “cm²”, “m²”, “ha”; ver *unit_convert*). Por defecto es kilómetros cuadrados “km²”.
- *perimeter_unit* = Puedes establecer una unidad de perímetro (por ejemplo, “km”, “cm”, “m”, “ha”; ver *unit_convert*). Por defecto es kilómetros “km”.
- *plot* = Genera histogramas básicos y un mapa de área núcleo - borde.
- *write* = Guarda la tabla (estadísticas del paisaje), el objeto sf (estadísticas de parches/nodos) y las gráficas. Es necesario especificar la ruta y el prefijo. Por ejemplo, para guardar en la ruta “C:/Folder” con el prefijo “Fragmentation”: “C:/Folder/Fragmentation”

```
MK_Fragmentation()
```

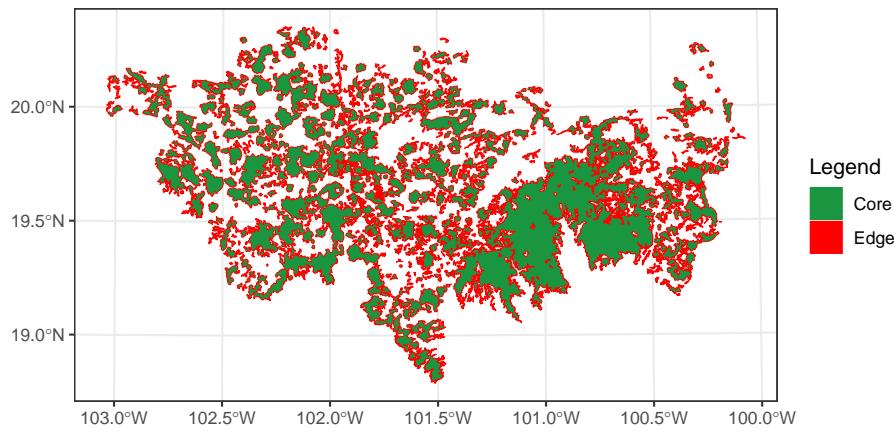
2.3 Ejercicio 1

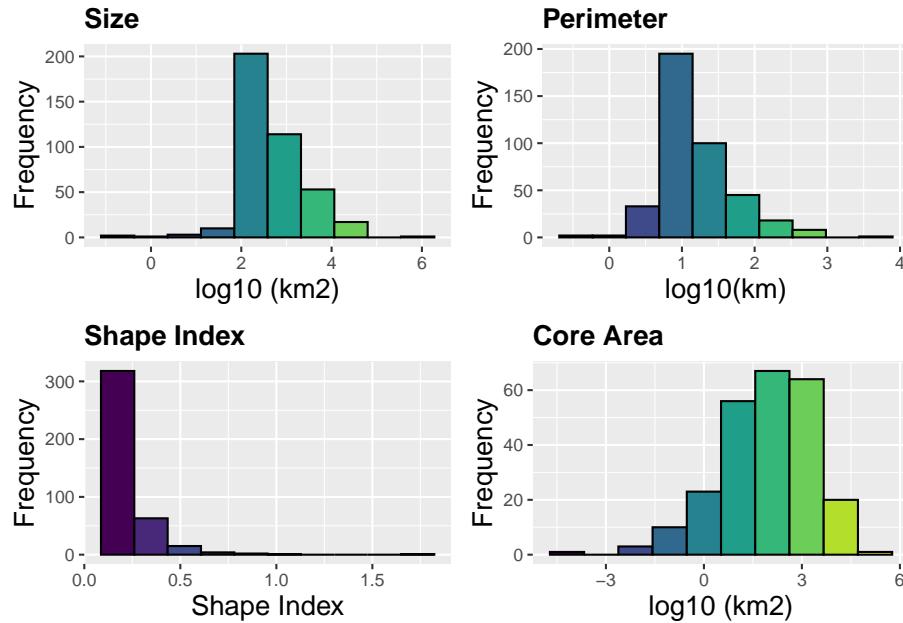
Estimamos el área del paisaje de estudio.

```
area_paisaje <- st_area(paisaje)
area_paisaje <- unit_convert(area_paisaje, "m2", "ha")
```

Aplicamos la función.

```
Fragmentacion <- MK_Fragmentation(nodes = habitat_nodes,
                                     edge_distance = 500,
                                     min_node_area = 100,
                                     landscape_area = area_paisaje,
                                     area_unit = "ha",
                                     perimeter_unit = "km",
                                     plot = TRUE)
```





2.3.1 Estadísticos a nivel de parche

El primer resultado “Patch statistics shapefile” es un shapefile con estadísticos de fragmentación a nivel de parche.

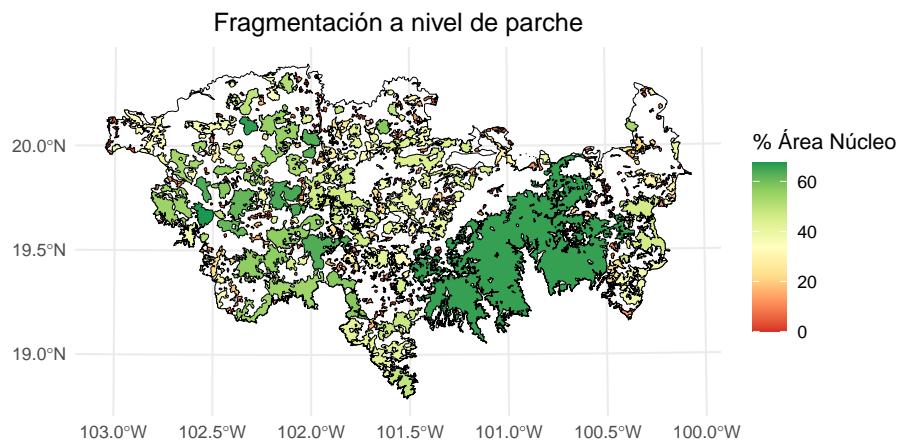
```
Fragmentacion$ Patch statistics shapefile
#> Simple feature collection with 404 features and 9 fields
#> Geometry type: POLYGON
#> Dimension: XY
#> Bounding box: xmin: -108954 ymin: 2025032 xmax: 202330.2 ymax: 2198936
#> Projected CRS: NAD_1927_Albers
#> First 10 features:
#>   Id      Area       CA CAPPercent Perimeter EdgePercent
#> 1  1  85.8368  0.000  0.0000    5.989  100.0000
#> 2  2 220.2168  0.000  0.0000   11.346  100.0000
#> 3  3 11019.9668 5337.795 48.4375  184.969  51.5625
#> 4  4 121.0018  0.000  0.0000    6.974  100.0000
#> 5  5 184.7226  0.000  0.0000   14.452  100.0000
#> 6  6  26.3052  0.000  0.0000    4.685  100.0000
#> 7  7  43.4931  0.000  0.0000    6.066  100.0000
#> 8  8  57.5414  0.000  0.0000    8.119  100.0000
#> 9  9 203.4670  0.000  0.0000   14.309  100.0000
#> 10 10 29440.4346 12326.275 41.8685  444.203  58.1315
#>           PARA ShapeIndex  FRAC
#>           geometry
```

```
#> 1 14.3324    0.1824 0.8040 POLYGON ((54911.05 2035815, ...
#> 2 19.4092    0.2157 0.9005 POLYGON ((44591.28 2042209, ...
#> 3 59.5774    0.4971 1.1217 POLYGON ((46491.11 2042467, ...
#> 4 17.3504    0.1788 0.8100 POLYGON ((54944.49 2048163, ...
#> 5 12.7818    0.3000 1.0235 POLYGON ((80094.28 2064140, ...
#> 6 5.6148     0.2577 0.9446 POLYGON ((69205.24 2066394, ...
#> 7 7.1700     0.2595 0.9557 POLYGON ((68554.2 2066632, ...
#> 8 7.0873     0.3019 1.0335 POLYGON ((69995.53 2066880, ...
#> 9 14.2195    0.2830 1.0012 POLYGON ((79368.68 2067324, ...
#> 10 66.2770   0.7303 1.1849 POLYGON ((23378.32 2067554, ...
```

Son espacialmente explicitos y podemos visualizarlos con librerias como ggplot2

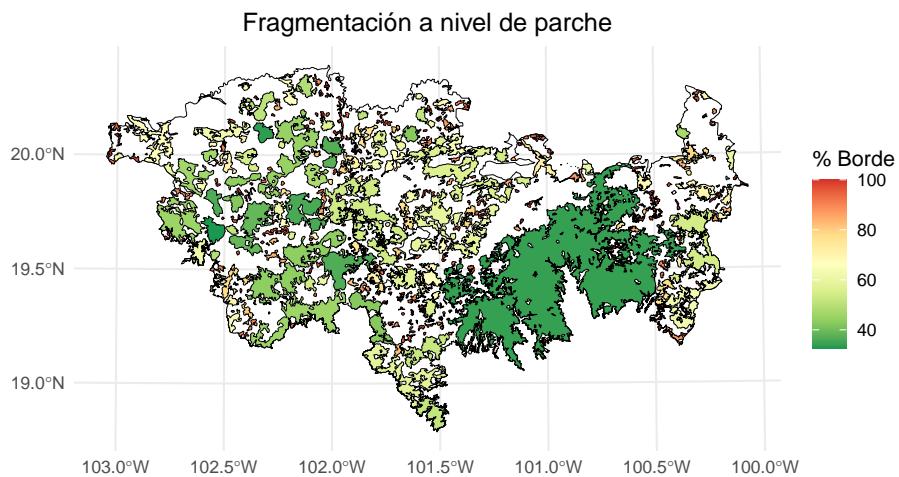
- % de área núcleo:

```
ggplot() +
  geom_sf(data = paisaje, aes(color = "Study area"), fill = NA, color = "black") +
  geom_sf(data = Fragmentacion$`Patch statistics shapefile`, aes(fill = CAPercent), color = "black")
  scale_fill_distiller(
    palette = "RdYlGn",
    direction = 1,
    name = "% Área Núcleo"
  ) +
  theme_minimal() +
  labs(
    title = "Fragmentación a nivel de parche",
    fill = "% Área Núcleo"
  ) +
  theme(
    legend.position = "right",
    plot.title = element_text(hjust = 0.5)
  )
```



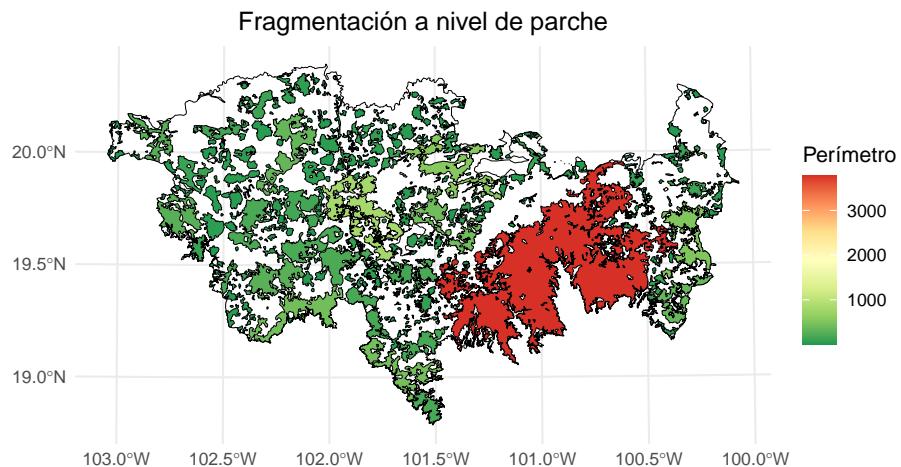
- % de borde

```
ggplot() +
  geom_sf(data = paisaje, aes(color = "Study area"), fill = NA, color = "black") +
  geom_sf(data = Fragmentacion$`Patch statistics shapefile`, aes(fill = EdgePercent),
  scale_fill_distiller(
    palette = "RdYlGn",
    direction = -1,
    name = "% Borde"
  ) +
  theme_minimal() +
  labs(
    title = "Fragmentación a nivel de parche",
    fill = "% Borde"
  ) +
  theme(
    legend.position = "right",
    plot.title = element_text(hjust = 0.5)
  )
```



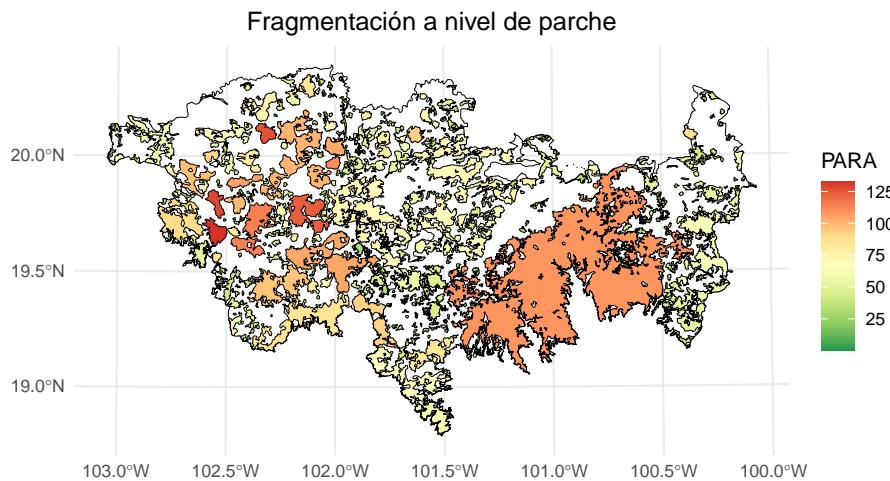
- Perimeter

```
ggplot() +
  geom_sf(data = paisaje, aes(color = "Study area"), fill = NA, color = "black") +
  geom_sf(data = Fragmentacion$`Patch statistics shapefile`, aes(fill = Perimeter), color = "black")
  scale_fill_distiller(
    palette = "RdYlGn",
    direction = -1,
    name = "Perímetro"
  ) +
  theme_minimal() +
  labs(
    title = "Fragmentación a nivel de parche",
    fill = "Perímetro"
  ) +
  theme(
    legend.position = "right",
    plot.title = element_text(hjust = 0.5)
  )
```



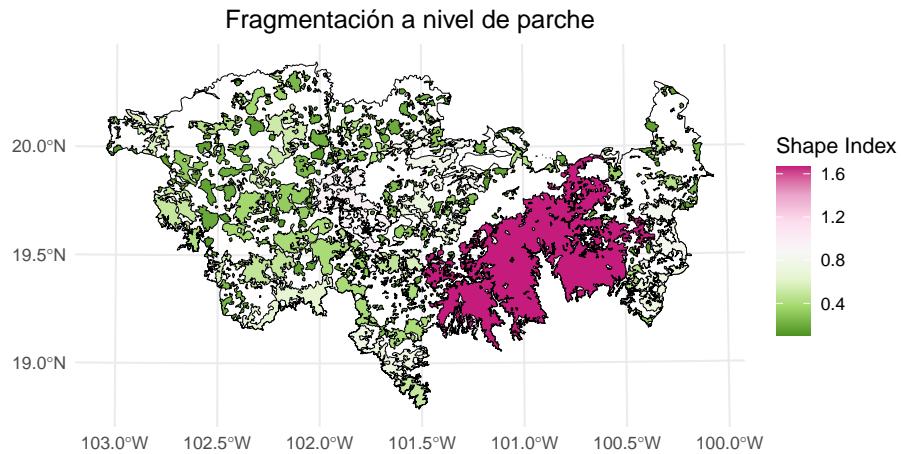
- Perimeter-Area Ratio

```
ggplot() +
  geom_sf(data = paisaje, aes(color = "Study area"), fill = NA, color = "black") +
  geom_sf(data = Fragmentacion$`Patch statistics shapefile`, aes(fill = PARA), color =
  scale_fill_distiller(
    palette = "RdYlGn",
    direction = -1,
    name = "PARA"
  ) +
  theme_minimal() +
  labs(
    title = "Fragmentación a nivel de parche",
    fill = "PARA"
  ) +
  theme(
    legend.position = "right",
    plot.title = element_text(hjust = 0.5)
  )
```



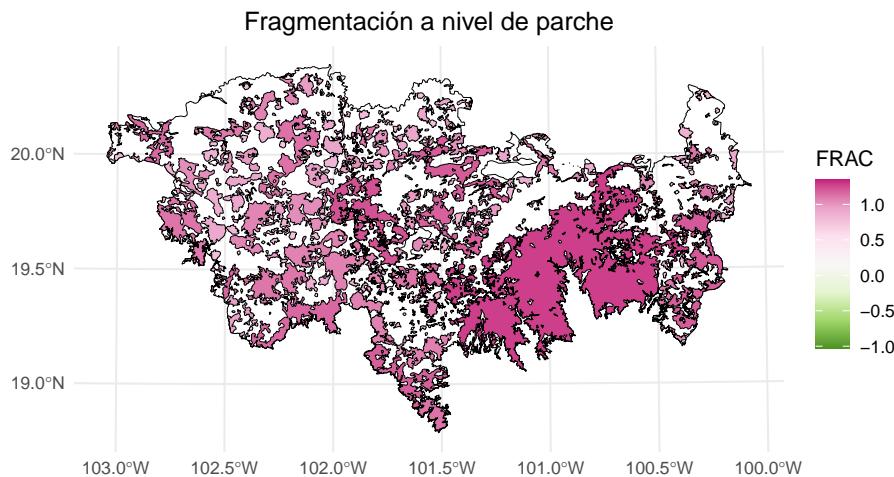
- Shape Index

```
ggplot() +
  geom_sf(data = paisaje, aes(color = "Study area"), fill = NA, color = "black") +
  geom_sf(data = Fragmentacion$`Patch statistics shapefile`, aes(fill = ShapeIndex), color = "black")
  scale_fill_distiller(
    palette = "PiYG",
    direction = -1,
    name = "Shape Index"
  ) +
  theme_minimal() +
  labs(
    title = "Fragmentación a nivel de parche",
    fill = "Shape Index"
  ) +
  theme(
    legend.position = "right",
    plot.title = element_text(hjust = 0.5)
  )
```



- Fractal Dimension Index

```
ggplot() +
  geom_sf(data = paisaje, aes(color = "Study area"), fill = NA, color = "black") +
  geom_sf(data = Fragmentacion$`Patch statistics shapefile`, aes(fill = FRAC), color =
  scale_fill_distiller(
    palette = "PiYG",
    direction = -1,
    name = "FRAC"
  ) +
  theme_minimal() +
  labs(
    title = "Fragmentación a nivel de parche",
    fill = "FRAC"
  ) +
  theme(
    legend.position = "right",
    plot.title = element_text(hjust = 0.5)
  )
```



2.3.2 Estadísticos a nivel de paisaje

Los resultados se presentan a manera de una lista, el primer resultado se llama “Summary landscape metrics (Viewer Panel)” y tiene estadisticos de fragmentación a nivel de paisaje.

```
class(Fragmentacion)
#> [1] "list"
Fragmentacion$`Summary landscape metrics (Viewer Panel)`
```

Metric

Value

Patch area (ha)

1273573.9249

Number of patches

404.0000

Size (mean)

3152.4107

Patches < minimum patch area

24 CHAPTER 2. ESTADÍSTICAS DE FRAGMENTACIÓN DEL PAISAJE

40.0000

Patches < minimum patch area (%)

0.1992

Total edge

17920.4740

Edge density

0.0141

Patch density

0.0146

Total Core Area (ha)

631595.3522

Cority

0.6064

Shape Index (mean)

0.2207

FRAC (mean)

0.8536

MESH (ha)

66572.5742

Métrica	Descripción	
Área de los fragmentos (km ²)	Suma del área de cada fragmento de vegetación presente en la unidad del paisaje (área protegida y su zona de influencia). Si el tamaño de los fragmentos es similar al área total de la unidad del paisaje se puede inferir que existe un bajo grado de fragmentación.	Cority
Número de fragmentos	Número de fragmentos de vegetación presentes en la unidad del paisaje.	Cority por fragmentación. Número de núcleo de 1 indica de 2002).
Tamaño de los fragmentos (km ²)	Área promedio los fragmentos de vegetación presentes en la unidad del paisaje.	En donde, número de número de
Fragmentos < 100 km ²	Número de fragmentos menores a 100 km ² .	Índice de forma
Fragmentos < 100 km ² (%)	Porcentaje del área de los fragmentos menores a 100 km ² (con respecto al área total de los fragmentos de vegetación).	Promedio vegetación compacto del fragmento 2002).
Borde total (m)	Longitud total (m) del borde de los fragmentos de vegetación en el paisaje.	
Densidad de borde (km ²)	Longitud del borde por km ² . Un valor de 0 se presenta cuando no existe borde en el paisaje. Esta medida facilita la comparación entre paisajes de diferentes tamaños. $DB = \frac{BT}{A}$ Donde, BT es el borde total y A el área de los fragmentos en el paisaje.	En donde, fragmento
Área núcleo (km ²)	Promedio del área núcleo. El área núcleo representa el área en el fragmento de hábitat que se encuentra a partir de una distancia de profundidad (borde) especificada desde el perímetro. La distancia de profundidad usada en este estudio fue de 500 m (Haddad et al., 2015).	
Dimensión fractal (FRAC)	Promedio de la dimensión fractal de los fragmentos de vegetación. El índice refleja la complejidad de la forma del fragmento. Una dimensión fractal mayor que 1 indica un aumento en la complejidad de la forma. Cuando el valor se aproxima a 1 la forma es simples, como los cuadrados (McGarigal et al., 2002). $FRAC = \frac{\sum_{i=1}^n \frac{2 \cdot \ln(0.25 \cdot Perímetro_i)}{\ln(A_i)}}{N}$ En donde A, es el área del fragmento y N es el número de fragmentos.	
MESH	Effective Mesh Size (MESH, por sus siglas en inglés) es una medida del grado de fragmentación en el paisaje que va de 0 al área total del paisaje. El MESH es máximo cuando el paisaje consiste en un solo fragmento de hábitat o el hábitat es continuo más allá de la unidad del paisaje analizada (Jaeger, 2000).	

La densidad de malla efectiva (MESH) es una medida del grado en que el movimiento entre diferentes partes del paisaje se ve interrumpido por una Fragmentación. El índice MESH se ha popularizado debido a su facilidad de interpretación. Si conocemos el área de nuestro paisaje, entonces podemos estimar el porcentaje de fragmentación:

```
mesh <- as.data.frame(Fragmentacion[[1]])
mesh <- mesh[13,2]
mesh_porcentaje <- (area_paisaje - mesh) * 100 / area_paisaje
mesh_porcentaje
#> [1] 97.58915
```

2.4 Ejercicio 2

Hagamos un loop en donde exploramos distintas distancias de profundidad del efecto borde.

```
#Loop edge distance
library(purrr)
Fragmentacion.2 <- map_dfr(seq(100, 1000, 100), function(x){
  x.1 <- MK_Fragmentation(nodes = habitat_nodes,
                            edge_distance = x, plot = FALSE)[[2]]
  CA <- mean(x.1$CAPercent)
  Edge <- mean(x.1$EdgePercent)
  x.2 <- rbind(data.frame('Edge distance' = x, Type = "Core Area", Percentage = CA),
                data.frame('Edge distance' = x, Type = "Edge", Percentage = Edge))
  return(x.2)
}, .progress = TRUE)
```

```
Fragmentacion.2
```

	Edge.distance	Type	Percentage
#> 1	100	Core Area	65.761207
#> 2	100	Edge	34.238793
#> 3	200	Core Area	41.980654
#> 4	200	Edge	58.019346
#> 5	300	Core Area	26.853211
#> 6	300	Edge	73.146789
#> 7	400	Core Area	18.110913
#> 8	400	Edge	81.889087
#> 9	500	Core Area	12.863545
#> 10	500	Edge	87.136455
#> 11	600	Core Area	9.554117
#> 12	600	Edge	90.445883
#> 13	700	Core Area	7.347448
#> 14	700	Edge	92.652552
#> 15	800	Core Area	5.767296
#> 16	800	Edge	94.232704

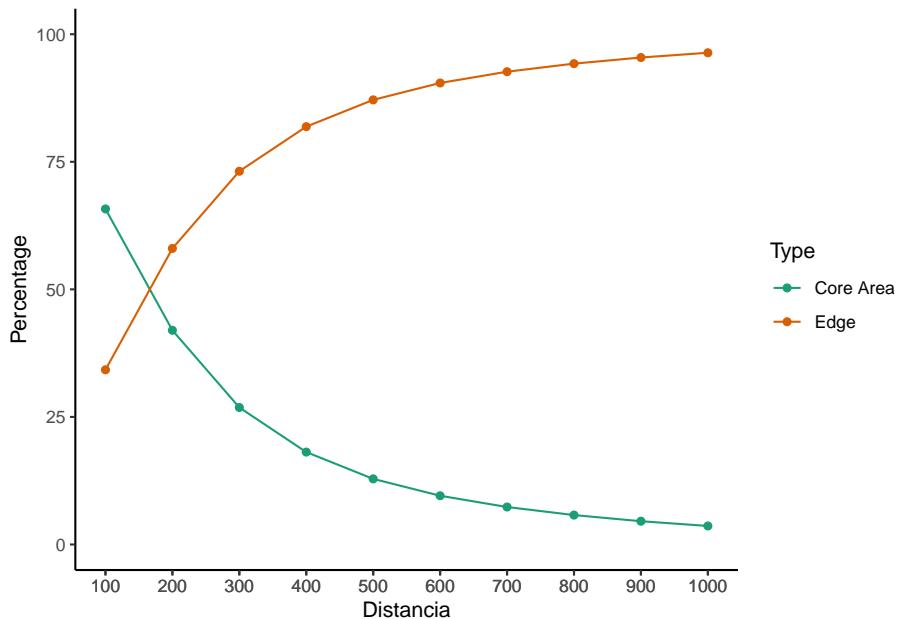
```
#> 17      900 Core Area  4.564680
#> 18      900     Edge  95.435320
#> 19     1000 Core Area  3.633780
#> 20     1000     Edge  96.366220
```

El porcentaje promedio de área núcleo (ausencia de efecto de borde) para todos los parches disminuye en más del 65% cuando se considera un efecto de borde con una penetración de 1 km.

Distancia de profundidad	%Área Núcleo
100	65.76%
500	12.86%
1000	3.63%

Podemos graficar el promedio del porcentaje de área núcleo de los parches y el porcentaje del borde de los parches (%área núcleo + % de borde = 100%).

```
library(ggplot2)
ggplot(Fragmentacion.2, aes(x=Edge.distance, y=Percentage, group=Type)) +
  geom_line(aes(color=Type))+
  geom_point(aes(color=Type))+ ylim(0,100) +
  scale_x_continuous("Distancia", labels = as.character(Fragmentacion.2$Edge.distance), breaks =
  scale_color_brewer(palette="Dark2")+
  theme_classic()
```

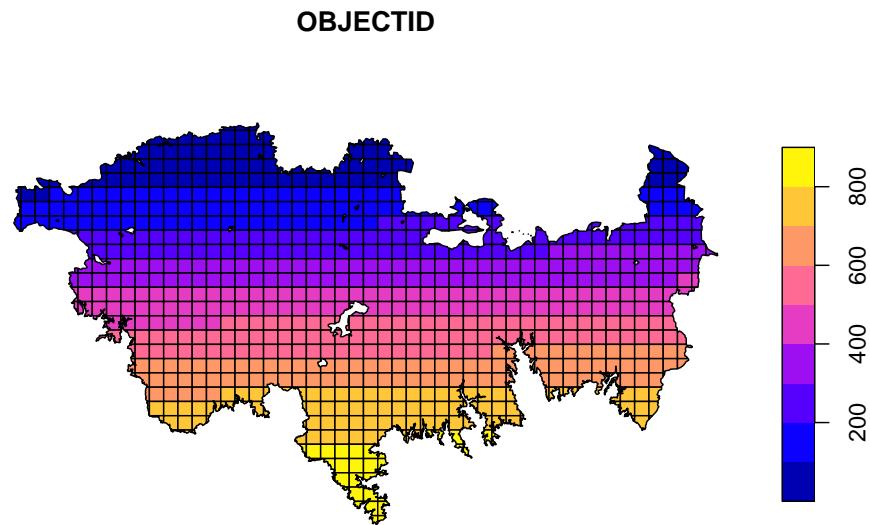


2.5 Ejercicio 3

Ahora probemos los alcances de la función estimando el índice MESH sobre un grid.

Grid de 40 km²

```
Grid_test <- make_grid(x = paisaje, hexagonal = FALSE,
                       cell_area = unit_convert(40, "km2", "m2"),
                       clip = TRUE)
plot(Grid_test)
```



Estimar MESH usando un loop sencillo tipo `for()`

```
#Variable dummy
Grid_test$MESH <- 0

for(i in 1:nrow(Grid_test)){
  cat(paste0(i, " de ", nrow(Grid_test), "\r"))
  grid.i <- Grid_test[i,]
  nodes.i <- suppressWarnings(st_intersection(habitat_nodes, grid.i))

  if(nrow(nodes.i) > 0){
    area_paisaje.i <- st_area(grid.i)
```

```

area_paisaje.i <- unit_convert(area_paisaje.i, "m2", "ha")
Fragmentacion.i <- MK_Fragmentation(nodes = nodes.i,
                                       edge_distance = 500,
                                       min_node_area = 100,
                                       landscape_area = area_paisaje.i,
                                       area_unit = "ha",
                                       perimeter_unit = "km",
                                       plot = FALSE)
mesh <- as.data.frame(Fragmentacion.i[[1]])
mesh <- mesh[13,2]
mesh_porcentage <-  (area_paisaje.i - mesh)*100/area_paisaje.i
Grid_test$MESH[i] <- mesh_porcentage
} else {
  Grid_test$MESH[i] <- 100
}
}
}

```

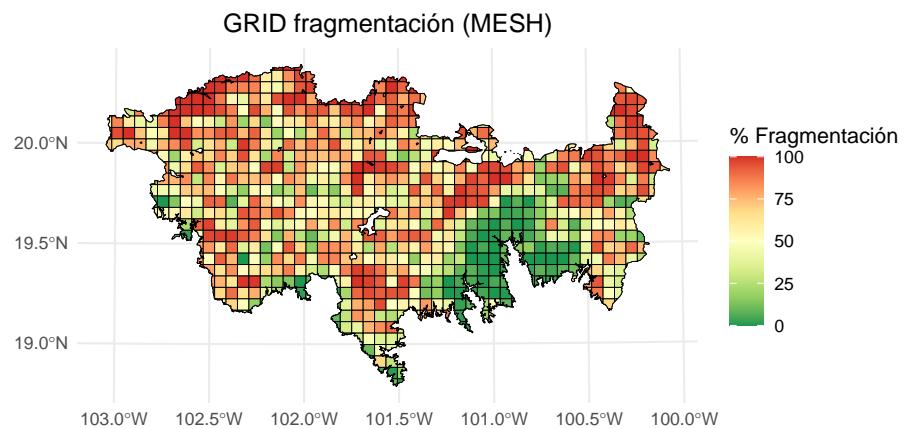
Lo podemos visualizar con ggplot2

```

ggplot() +
  geom_sf(data = paisaje, aes(color = "Study area"), fill = NA, color = "black") +
  geom_sf(data = Grid_test, aes(fill = MESH), color = "black", size = 0.1) +
  scale_fill_distiller(
    palette = "RdYlGn",
    direction = -1,
    name = "% Fragmentación"
  ) +
  theme_minimal() +
  labs(
    title = "GRID fragmentación (MESH)",
    fill = "% Fragmentación"
  ) +
  theme(
    legend.position = "right",
    plot.title = element_text(hjust = 0.5)
  )

```

30 CHAPTER 2. ESTADÍSTICAS DE FRAGMENTACIÓN DEL PAISAJE



Chapter 3

Índices de centralidad

3.1 Insumos y paquetes

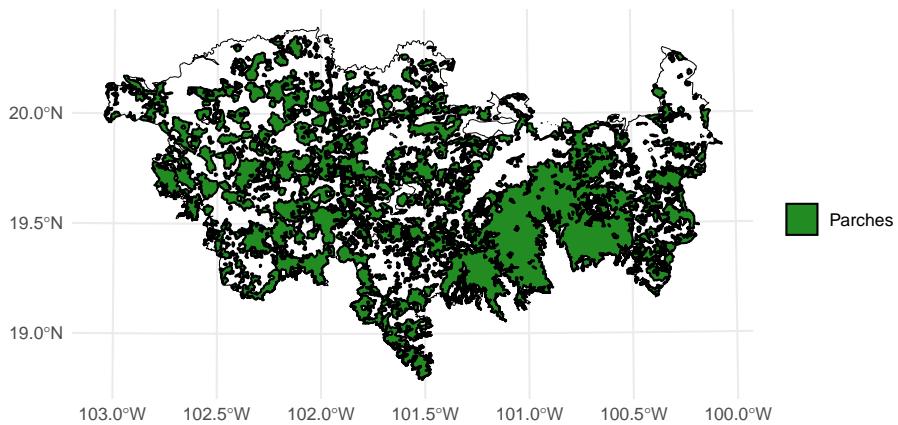
Seguimos trabajando con los mismos shapefiles de la sección anterior: habitat_nodes y paisaje.

```
library(ggplot2)
library(sf)
library(Makurhini)
library(RColorBrewer)

#> Cargando paquete requerido: igraph
#>
#> Adjuntando el paquete: 'igraph'
#> The following objects are masked from 'package:stats':
#>
#>     decompose, spectrum
#> The following object is masked from 'package:base':
#>
#>     union
#> Cargando paquete requerido: cppRouting
#> Linking to GEOS 3.13.0, GDAL 3.10.1, PROJ 9.5.1;
#> sf_use_s2() is TRUE
#> [1] 404

ggplot() +
  geom_sf(data = paisaje, aes(color = "Study area"), fill = NA, color = "black") +
  geom_sf(data = habitat_nodes, aes(color = "Parches"), fill = "forestgreen", linewidth = 0.5) +
  scale_color_manual(name = "", values = "black") +
```

```
theme_minimal() +
  theme(axis.title.x = element_blank(),
        axis.title.y = element_blank())
```



En caso de necesitar abrir otro vector (e.g., .shp, .gpkg) necesitan usar la función `read_sf()` del paquete `sf`, la función `shapefile()` del paquete `raster`, o la función `vect()` del paquete `terra`.

Para abrirlo solo necesitan colocar la dirección de su archivo, el nombre y la extensión, ejemplo:

- `vegetation_patches <- sf::read_sf("D:/Datos/vegetation_patches.shp")`
- `vegetation_patches <- raster::shapefile("D:/Datos/vegetation_patches.shp")`
- `vegetation_patches <- terra::vect("D:/Datos/vegetation_patches.shp")`

3.2 MK_RMCentrality()

La función `MK_RMCentrality()` calcula medidas radiales (es decir, grado, fuerza, centralidad de vectores propios y centralidad de proximidad) y mediales (es decir, centralidad de interrelación, pertenencia a nodos y modularidad) de la centralidad de nodos para identificar, por ejemplo, stepping stones.

Las medidas o índices que estima son los siguientes:

3.2.1 1. Centralidad de Grado (degree)

Cuántas conexiones directas tiene un nodo.

- Es como contar cuántos parches de hábitat o áreas protegidas están conectados con cada parche.
- Más conexiones = mayor grado = más centralidad.

3.2.2 2. Fuerza (strength) (*para redes ponderadas*)

Como el grado, pero suma los **pesos** (o probabilidades) de los enlaces.

- En lugar de solo contar conexiones, se suman qué tan **fuertes o probables** son (por ejemplo, probabilidad de dispersión).
- Un nodo con pocas pero fuertes conexiones puede ser más central que uno con muchas pero débiles.

3.2.3 3. Centralidad de Vector Propio (eigen)

Mide cuán conectado está un nodo con **otros nodos importantes para la conectividad**.

- Un nodo es importante si está conectado a otros nodos que también lo son.
- Útil para detectar “nodos influyentes” en la red.

3.2.4 4. Centralidad de Cercanía (close)

Qué tan cerca está un nodo de todos los demás.

- Se calcula como el inverso de la suma de distancias a todos los demás nodos.
 - Los nodos con alta cercanía pueden **difundir o recibir flujo rápidamente**.
-

3.2.5 Medidas Mediales (*¿Quién está en medio o conecta otros nodos?*)

3.2.5.1 5. Centralidad de Intermediación (BWC)

Cuántas veces un nodo se encuentra **en las rutas más cortas** entre otros nodos.

- Nodos con alta intermediación actúan como puentes (**stepping stones** o **cuellos de botella**).
 - Muy importantes para la conectividad — si se eliminan, pueden fragmentar la red.
-

3.2.6 Detección de Comunidades (*¿Quién pertenece al mismo grupo?*)

3.2.6.1 6. Caminatas Aleatorias Cortas (`memb.rw`)

Agrupa nodos según la probabilidad de que una caminata aleatoria **permanezca dentro del grupo**.

- Simula un animal moviéndose aleatoriamente por la red.
- Detecta grupos **fuertemente conectados**.

3.2.6.2 7. Algoritmo de Louvain (`memb.louvain`)

Divide la red en comunidades para **maximizar la modularidad** (qué tan separados están los grupos).

- Encuentra grupos donde los nodos están más conectados entre sí que con el resto.
 - Es rápido y muy usado en redes ecológicas grandes.
-

3.2.7 La función (no correr)

```
MK_RMCentrality(
  nodes,
  distance = list(type = "centroid"),
  distance_thresholds = NULL,
  binary = TRUE,
  probability = NULL,
  rasterparallel = FALSE,
  write = NULL,
  intern = TRUE
)
```

3.2.8 Descripción de los argumentos de la función

Argumento	Tipo	Descripción
<code>nodes</code>	objeto	Objeto que representa los nodos o fragmentos de hábitat. Puede ser un <code>data.frame</code> , objeto espacial vectorial (<code>sf</code> , <code>SpatVector</code> , etc.) o raster (<code>RasterLayer</code> , <code>SpatRaster</code>). Debe estar en un sistema de coordenadas proyectadas. En rasters, los valores deben ser enteros (ID de los nodos) y los no hábitat como NA.

Argumento	Tipo	Descripción
<code>distance</code>	matriz o lista	Matriz cuadrada con las distancias entre nodos o una lista con los parámetros para calcularlas. Puede incluir tipo ("centroid", "edge", "least-cost", "commute-time") y resistencia.
<code>distance_thresholds</code>	numérico	Distancia de dispersión (en metros) de la especie. Si es NULL, se calcula como la mediana entre nodos. También se puede estimar con la función <code>dispersal_distance</code> .
<code>binary</code>	lógico	Si es TRUE, se calcula conectividad binaria: nodos conectados (1) o no conectados (0) según el umbral de distancia. No usa <code>probability</code> .
<code>probability</code>	numérico	Probabilidad de conexión asociada al umbral de distancia. Por ejemplo, 0.5 para distancias medianas, 0.05 para distancias máximas. Por defecto es 0.5 si es NULL.

Argumento	Tipo	Descripción
<code>rasterparallel</code>	lógico	Si <code>nodes</code> es un raster, permite asignar las métricas calculadas al raster de nodos. Útil cuando la resolución es menor a 100 m ² .
<code>write</code>	texto	Ruta y prefijo para guardar los resultados (<code>sf</code>). Ejemplo: "C:/ejemplo".
<code>intern</code>	lógico	Muestra el progreso del proceso. Por defecto <code>TRUE</code> . Puede no llegar al 100% si el cálculo es muy rápido.

3.3 Ejemplo 1

```
library(Makurhini)
library(sf)
data("habitat_nodes", package = "Makurhini")
nrow(habitat_nodes) # Number of patches
#> [1] 404
#Two distance threshold,
centrality_test <- MK_RMCentrality(nodes = habitat_nodes,
                                      distance = list(type = "centroid"),
                                      distance_thresholds = 10000,
                                      probability = 0.5,
                                      write = NULL)
#> Done!
centrality_test
#> Simple feature collection with 404 features and 8 fields
#> Geometry type: POLYGON
#> Dimension: XY
#> Bounding box: xmin: -108954 ymin: 2025032 xmax: 202330.2 ymax: 2198936
#> Projected CRS: NAD_1927_Albers
#> First 10 features:
#>   Id strength      eigen      close    BWC cluster memb.rw
```

```
#> 1 1 30228524 0.0010435836 0.03840356 0 1 6
#> 2 2 21600031 0.0006195356 0.03995935 1 1 6
#> 3 3 29320545 0.0009026418 0.03831019 0 1 6
#> 4 4 16499522 0.0005867564 0.04187906 23 1 6
#> 5 5 26068911 0.0011987437 0.04240465 0 1 6
#> 6 6 12737692 0.0005630043 0.04627714 17 1 6
#> 7 7 12497243 0.0005499038 0.04634291 15 1 6
#> 8 8 13198398 0.0005859873 0.04607765 0 1 6
#> 9 9 22276433 0.0010276194 0.04296412 567 1 6
#> 10 10 12804425 0.0004306005 0.04405417 1063 1 6
#> memb.louvain geometry
#> 1
#> 2
#> 3
#> 4
#> 5
#> 6
#> 7
#> 8
#> 9
#> 10
```

Exploraremos otra forma de hacer el plot usando intervalos

```
install.packages("ClassInt"), dependencies = TRUE
install.packages("dplyr"), dependencies = TRUE
```

- Strength:

```
library(classInt)
library(dplyr)
#>
#> Adjuntando el paquete: 'dplyr'
#> The following objects are masked from 'package:igraph':
#>
#>     as_data_frame, groups, union
#> The following objects are masked from 'package:stats':
#>
#>     filter, lag
#> The following objects are masked from 'package:base':
#>
#>     intersect, setdiff, setequal, union

# Calcular los intervalos de Jenks para strength
```

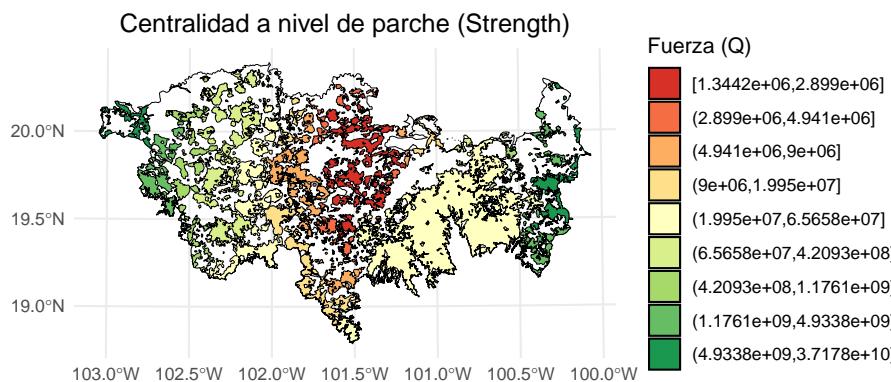
```

breaks <- classInt::classIntervals(centrality_test$strength, n = 9, style = "quantile")

# Crear una nueva variable categórica con los intervalos
centrality_test <- centrality_test %>%
  mutate(strength_q = cut(strength,
                          breaks = breaks$brks,
                          include.lowest = TRUE,
                          dig.lab = 5))

# Graficar en ggplot2 usando las clases Jenks
ggplot() +
  geom_sf(data = paisaje, fill = NA, color = "black") +
  geom_sf(data = centrality_test, aes(fill = strength_q), color = "black", size = 0.1) +
  scale_fill_brewer(palette = "RdYlGn", direction = 1, name = "Fuerza (Q)") +
  theme_minimal() +
  labs(
    title = "Centralidad a nivel de parche (Strength)",
    fill = "Strength\n(Jenks)"
  ) +
  theme(
    legend.position = "right",
    plot.title = element_text(hjust = 0.5)
)

```

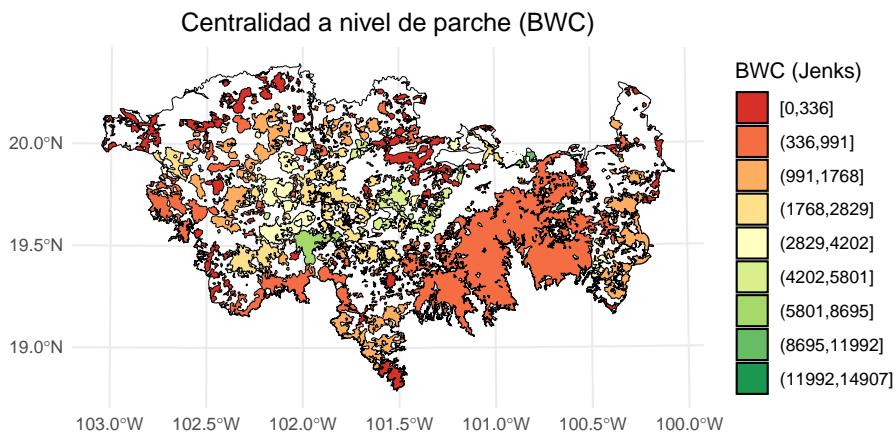


- BWC:

```

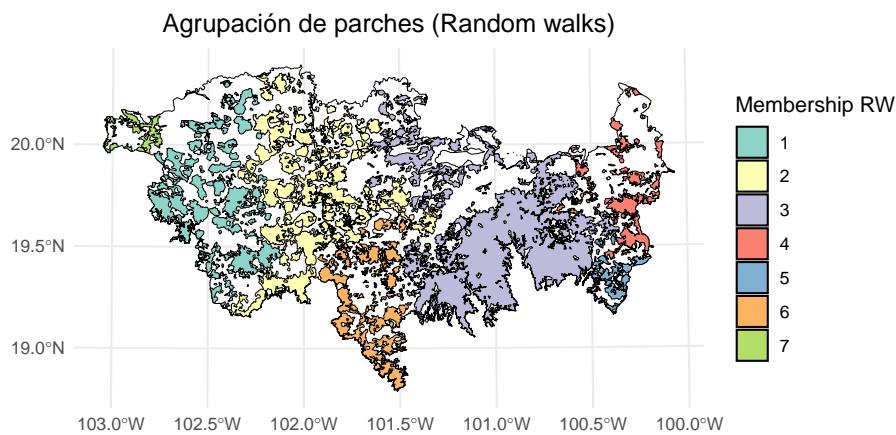
breaks <- classInt::classIntervals(centrality_test$BWC, n = 9, style = "jenks")
centrality_test <- centrality_test %>%
  mutate(BWC_jenks = cut(BWC,
    breaks = breaks$brks,
    include.lowest = TRUE,
    dig.lab = 5))
ggplot() +
  geom_sf(data = paisaje, fill = NA, color = "black") +
  geom_sf(data = centrality_test, aes(fill = BWC_jenks), color = "black", size = 0.1) +
  scale_fill_brewer(palette = "RdYlGn", direction = 1, name = "BWC (Jenks)") +
  theme_minimal() +
  labs(
    title = "Centralidad a nivel de parche (BWC)",
    fill = "BWC\n(Jenks)"
  ) +
  theme(
    legend.position = "right",
    plot.title = element_text(hjust = 0.5)
)

```



- Membresía por random walk:

```
ggplot() +
  geom_sf(data = paisaje, fill = NA, color = "black") +
  geom_sf(data = centrality_test, aes(fill = as.factor(memb.rw)), color = "black", size = 0.1) +
  scale_fill_brewer(
    palette = "Set3",
    name = "Membership RW"
  ) +
  theme_minimal() +
  labs(
    title = "Agrupación de parches (Random walks)",
    fill = "Membership RW"
  ) +
  theme(
    legend.position = "right",
    plot.title = element_text(hjust = 0.5)
  )
)
```



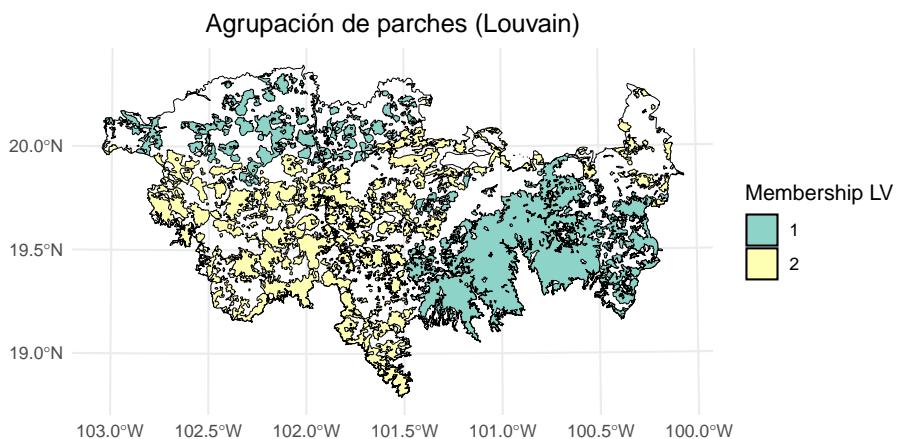
- Membresía por Louvain:

```
ggplot() +
  geom_sf(data = paisaje, fill = NA, color = "black") +
  geom_sf(data = centrality_test, aes(fill = as.factor(memb.louvain)), color = "black", size = 0.1) +
  scale_fill_brewer(
```

```

    palette = "Set3",
    name = "Membership LV"
) +
theme_minimal() +
labs(
  title = "Agrupación de parches (Louvain)",
  fill = "Membership LV"
) +
theme(
  legend.position = "right",
  plot.title = element_text(hjust = 0.5)
)

```



3.4 Ejemplo 2

Usando más de un umbral de distancia

```

centrality_test <- MK_RMCentrality(nodes = habitat_nodes,
                                      distance = list(type = "centroid"),
                                      distance_thresholds = c(10000, 100000),
                                      probability = 0.5,
                                      write = NULL)

```

```

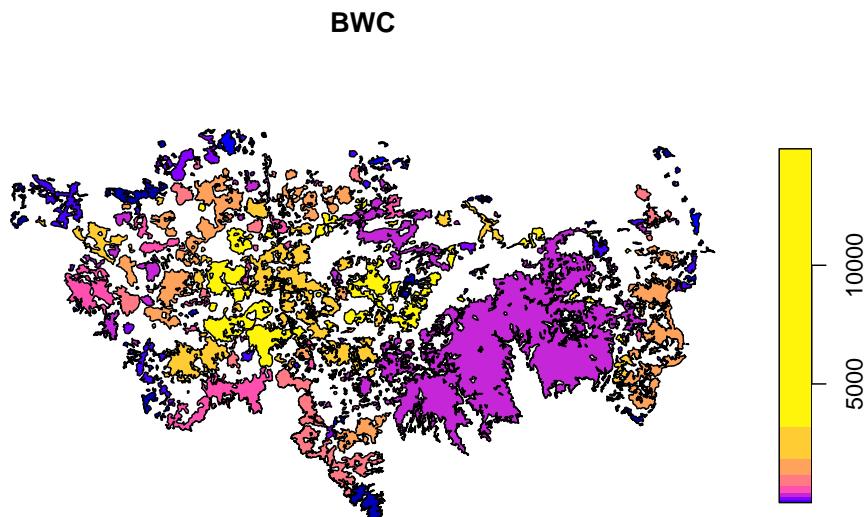
#> ======>----- 50% | ETA: 4s
#> Done!
centrality_test
#> $d10000
#> Simple feature collection with 404 features and 8 fields
#> Geometry type: POLYGON
#> Dimension: XY
#> Bounding box: xmin: -108954 ymin: 2025032 xmax: 202330.2 ymax: 2198936
#> Projected CRS: NAD_1927_Albers
#> First 10 features:
#>   Id strength      eigen    close  BWC cluster memb.rw
#> 1  1 30228524 0.0010435836 0.03840356    0      1      6
#> 2  2 21600031 0.0006195356 0.03995935    1      1      6
#> 3  3 29320545 0.0009026418 0.03831019    0      1      6
#> 4  4 16499522 0.0005867564 0.04187906   23      1      6
#> 5  5 26068911 0.0011987437 0.04240465    0      1      6
#> 6  6 12737692 0.0005630043 0.04627714   17      1      6
#> 7  7 12497243 0.0005499038 0.04634291   15      1      6
#> 8  8 13198398 0.0005859873 0.04607765    0      1      6
#> 9  9 222716433 0.0010276194 0.04296412  567      1      6
#> 10 10 12804425 0.0004306005 0.04405417 1063      1      6
#>   memb.louvain           geometry
#> 1          1 POLYGON ((54911.05 2035815, ...
#> 2          1 POLYGON ((44591.28 2042209, ...
#> 3          1 POLYGON ((46491.11 2042467, ...
#> 4          1 POLYGON ((54944.49 2048163, ...
#> 5          2 POLYGON ((80094.28 2064140, ...
#> 6          2 POLYGON ((69205.24 2066394, ...
#> 7          2 POLYGON ((68554.2 2066632, ...
#> 8          2 POLYGON ((69995.53 2066880, ...
#> 9          2 POLYGON ((79368.68 2067324, ...
#> 10         1 POLYGON ((23378.32 2067554, ...
#>
#> $`d1e+05`
#> Simple feature collection with 404 features and 8 fields
#> Geometry type: POLYGON
#> Dimension: XY
#> Bounding box: xmin: -108954 ymin: 2025032 xmax: 202330.2 ymax: 2198936
#> Projected CRS: NAD_1927_Albers
#> First 10 features:
#>   Id strength      eigen    close  BWC cluster memb.rw
#> 1  1 958.4815 0.6679253 0.4207380    0      1      1
#> 2  2 925.0415 0.6471542 0.4356561    0      1      1
#> 3  3 960.2533 0.6700437 0.4197704    0      1      1
#> 4  4 896.6891 0.6266304 0.4494531    0      1      1

```

```
#> 5 5 871.2094 0.6057986 0.4632174 0 1 1
#> 6 6 836.4790 0.5842254 0.4818295 0 1 1
#> 7 7 836.5028 0.5843345 0.4818097 0 1 1
#> 8 8 837.6158 0.5848721 0.4811889 0 1 1
#> 9 9 858.0974 0.5971238 0.4701554 0 1 1
#> 10 10 874.7611 0.6125436 0.4606972 0 1 1
#> memb.louvain geometry
#> 1 1 POLYGON ((54911.05 2035815, ...
#> 2 1 POLYGON ((44591.28 2042209, ...
#> 3 1 POLYGON ((46491.11 2042467, ...
#> 4 1 POLYGON ((54944.49 2048163, ...
#> 5 1 POLYGON ((80094.28 2064140, ...
#> 6 1 POLYGON ((69205.24 2066394, ...
#> 7 1 POLYGON ((68554.2 2066632, ...
#> 8 1 POLYGON ((69995.53 2066880, ...
#> 9 1 POLYGON ((79368.68 2067324, ...
#> 10 1 POLYGON ((23378.32 2067554, ...
```

10 km:

```
plot(centrality_test$d10000["BWC"], breaks = "quantile")
```



100 km:

```
plot(centrality_test`^d1e+05`["BWC"], breaks = "quantile")
```

