Análisis de la Conectividad Ecológica en la Caatinga

Alumno: Cristian Retete

ÍNDICE

1. Introducción ... 3
2. Metodología ... 4
3. Resultados ... 4
Figura 1. Parches del área de Caatinga ... 5
Figura 2. Métricas del área de Caatinga ... 5
Figura 3. Fragmentación a nivel de parche del área de Caatinga ... 6
Figura 4. Centralidad a nivel de parche (BWC) ... 6
Figura 5. Agrupación de parches (Random walks) ... 7
Figura 6. Agrupación de parches (Louvain) ... 7
Figura 7. Índice BWC ... 8
Figura 8. Índice dICC ... 8
Figura 9. Índice dIICIntra ... 9
Figura 10. Índice dIICFlux ... 9
Figura 11. Índice dIICConnector ... 10
Figura 12. Índice dPC ... 10
Figura 13. Índice dPCIntra ... 11
Figura 14. Índice dPCFlux ... 11
Figura 15. Índice dPCConnector ... 12
Figura 16. Parches de restauración ... 12
Figura 17. Áreas prioritarias de restauración. ... 13
Figura 18. Datos cuantitativos de 683 fragmentos del paisaje analizados mediante múltiples índices de conectividad. ... 13
Figura 19. Probabilidad de conectividad en paisajes focales. ... 14
Figura 20. Índice de conectividad compuesta. ... 14
Figura 21. Corredores potenciales. ... 15
Figura 22. Priorización de enlaces. ... 15
4. Discusión ... 16
5. Conclusiones ... 17
6. Bibliografía ... 18
7. Anexo: Script en R utilizado ... 20

1. Introducción

La ecorregión Caatinga, ubicada en el noreste de Brasil, es el bioma semiárido más extenso de América del Sur, caracterizado por una gran heterogeneidad ecológica y una elevada biodiversidad con alto grado de endemismo (Silva et al., 2017). Este bioma es fundamental para la provisión de servicios ecosistémicos como regulación hídrica, almacenamiento de carbono y hábitat para especies amenazadas (Vieilledent et al., 2022).

En las últimas décadas, la Caatinga ha enfrentado una creciente presión antrópica debido a la expansión agrícola, el pastoreo intensivo, la deforestación y la urbanización, lo que ha conducido a una pérdida acelerada de cobertura vegetal y fragmentación del paisaje (Hansen et al., 2013; Potapov et al., 2020). Estos procesos han reducido la conectividad ecológica, limitando el flujo génico, los movimientos de fauna y la resiliencia del ecosistema frente a perturbaciones naturales y antrópicas (Saura y Pascual-Hortal, 2007; Haddad et al., 2015).

La conectividad ecológica es esencial para mantener la funcionalidad de los ecosistemas. En paisajes fragmentados, los corredores biológicos actúan como puentes que facilitan la dispersión de organismos y reducen los efectos negativos del aislamiento (Crooks y Sanjayan, 2006). El análisis de conectividad estructural, mediante índices como el Índice Integral de Conectividad focal (IICf), la Probabilidad de Conectividad focal (PCf) y el Índice de Conectividad Compuesto (CCIf), permite identificar áreas críticas para la conservación y restauración ecológica (Pascual-Hortal y Saura, 2006).

Este estudio tiene como objetivo evaluar el estado actual y los cambios en la conectividad ecológica de la vegetación primaria en la Caatinga, identificando nodos y enlaces prioritarios que guíen estrategias de manejo sostenible. Los resultados aportan información científica clave para la planificación territorial y la formulación de políticas públicas orientadas a la conservación de este bioma único.

2. Metodología

El área de estudio corresponde a la ecorregión Caatinga, localizada en el noreste de Brasil, caracterizada por clima semiárido, precipitaciones estacionales y vegetación xerofítica. El análisis se centró en la vegetación primaria para evaluar la conectividad estructural del paisaje. Fuentes de datos geoespaciales utilizadas:

- Huella espacial humana: Dryad, SEDAC-NASA, Zenodo.
- Ecorregiones: https://ecoregions.appspot.com/
- Vegetación: Hansen et al. (2013), Potapov et al. (2020), ESA Land Cover, MODIS MCD12Q1, Dynamic World, MapBiomas, GLanCE30 v001 30m, CONABIO (México).

Se aplicaron métricas de conectividad: IICf, PCf y CCIf, para evaluar la importancia relativa de cada parche de vegetación. Además, se priorizaron enlaces para identificar conexiones clave cuya pérdida afectaría la conectividad global. El procesamiento se realizó en R, empleando herramientas análisis espacial y de grafos.

Para el análisis de datos se utilizó la herramienta Makurhini, que es un paquete de R diseñado para calcular índices de fragmentación y conectividad de paisajes. Permite evaluar redes de áreas protegidas y medir la importancia de distintos elementos del paisaje para mantener la conectividad ecológica.

3. Resultados

Se presentan las figuras resultantes del análisis. Debajo de cada figura se ofrece una descripción breve y concisa.

3.1. Índices de fragmentación (número de fragmentos, tamaño medio)

En la figura 1 y 3 se puede observar Número de fragmentos alto y tamaño medio reducido en amplias zonas; esto indica una fragmentación severa que afecta la continuidad del hábitat y aumenta la vulnerabilidad de especies. Con 683 parches en el ares y una media de 126,89 (ver figura 2).

Figura 1. Parches del área de Caatinga

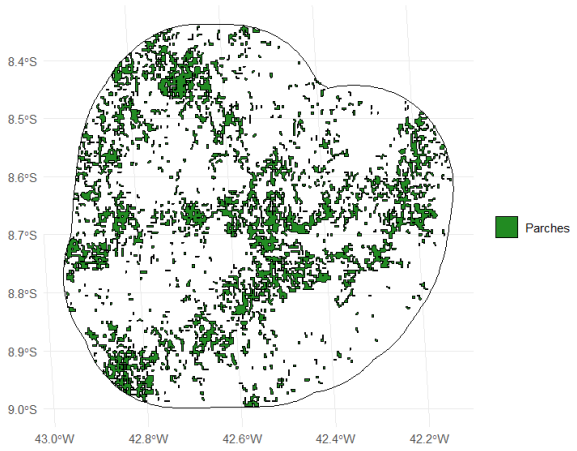


Figura 2. Métricas del área de Caatinga

Metric	Value
Patch area (ha)	86670.1275
Number of patches	683.0000
Size (mean)	126.8962
Patches < minimum patch area	582.0000
Patches < minimum patch area (%)	20.1907
Total edge	4692.4460
Edge density	0.0541
Patch density	0.7880
Total Core Area (ha)	388.2465
Cority	0.0176
Shape Index (mean)	0.1588
FRAC (mean)	0.5811
MESH (ha)	3202.3183

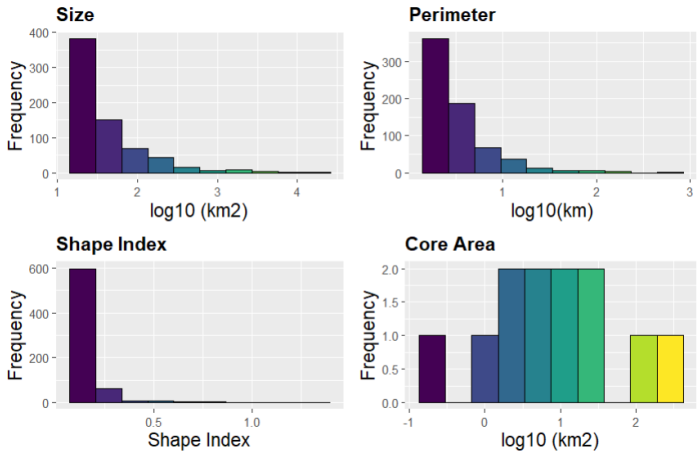
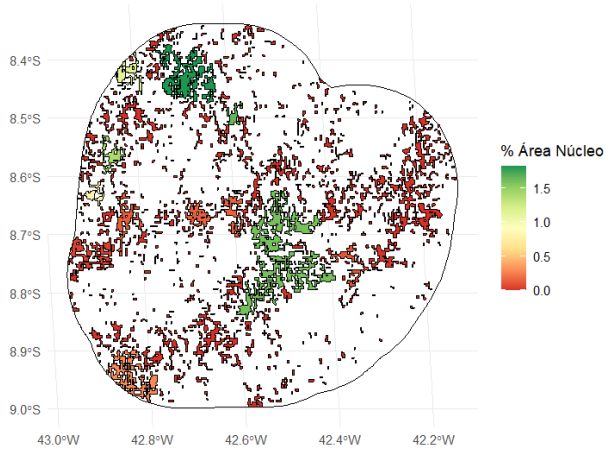


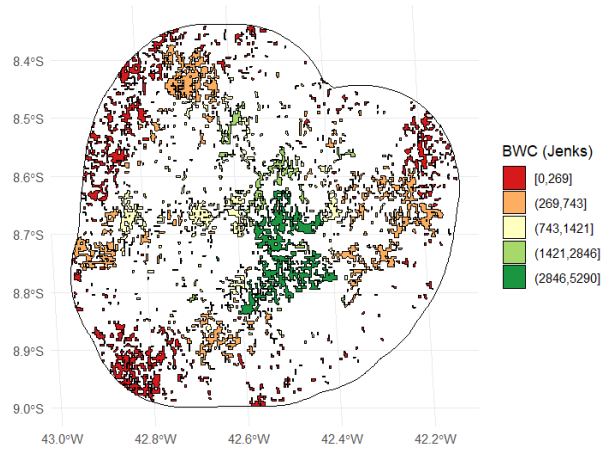
Figura 3. Fragmentación a nivel de parche del área de Caatinga



3.2. Centralidad (degree, BWC, etc.)

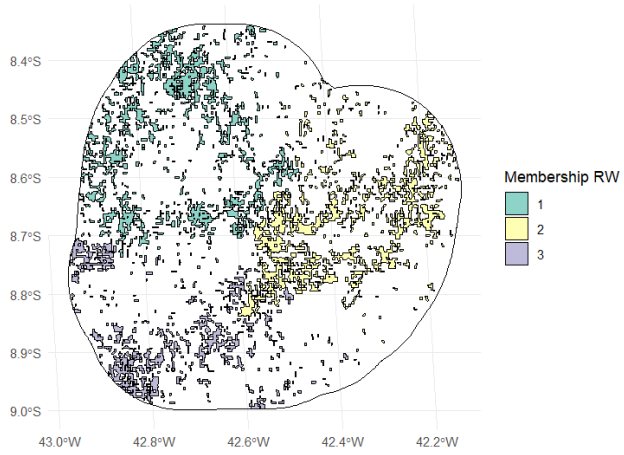
Esta imagen (figura 4), la Centralidad de Intermediación a nivel de parche (BWC), donde los fragmentos verdes en el centro tienen los valores más altos (2846-5290) y actúan como conectores críticos del paisaje, mientras que las áreas rojas periféricas presentan valores bajos (0-269) indicando mayor aislamiento. El patrón revela un núcleo central altamente conectado rodeado por fragmentos progresivamente más aislados hacia la periferia. Los fragmentos centrales son prioritarios para conservación ya que su pérdida fragmentaría severamente la conectividad del paisaje.

Figura 4. Centralidad a nivel de parche (BWC)



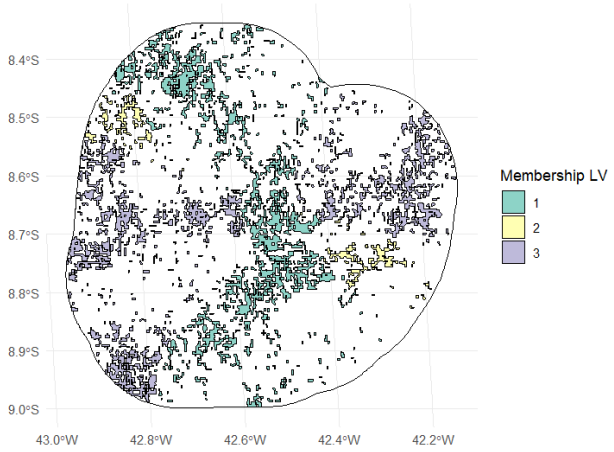
La (figura 5), se logra identificar tres comunidades de fragmentos conectados: el grupo 1 (verde claro) domina el norte y oeste, el grupo 2 (amarillo) se concentra en el centro-sur, y el grupo 3 (azul-morado) aparece disperso. Esta clasificación revela la estructura modular del paisaje, donde cada grupo representa fragmentos más conectados entre sí que con otros grupos.

Figura 5. Agrupación de parches (Random walks)



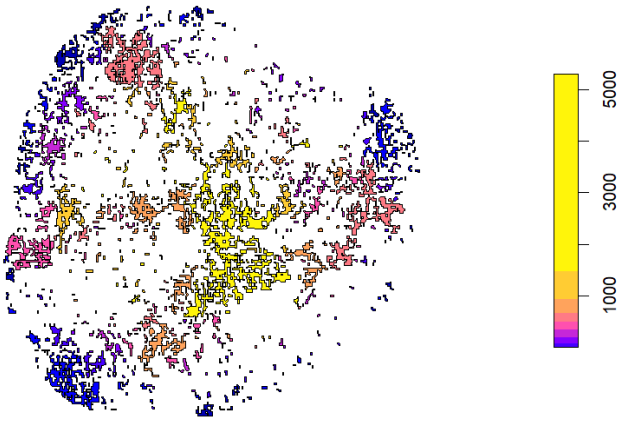
La (figura 6), muestra la Agrupación de parches mediante el algoritmo de Louvain, donde se identifica tres comunidades de fragmentos basándose en la optimización de la modularidad del paisaje. El grupo 1 (verde claro) se distribuye principalmente en el noroeste y centro, el grupo 2 (amarillo) forma clusters en el centro-este, y el grupo 3 (azul-morado) aparece disperso por todo el paisaje. Comparado con Random Walks, el método de Louvain genera una agrupación diferente que puede revelar patrones de conectividad más complejos y optimizados para la estructura modular del paisaje.

Figura 6. Agrupación de parches (Louvain)



La (figura 7), la Centralidad de Intermediación (BWC) donde los fragmentos amarillos centrales tienen los valores más altos de conectividad (hasta 5000) actuando como conectores críticos, mientras que los fragmentos azules y morados periféricos presentan baja conectividad. La visualización revela una estructura radial con un núcleo central altamente conectado rodeado de fragmentos más aislados.

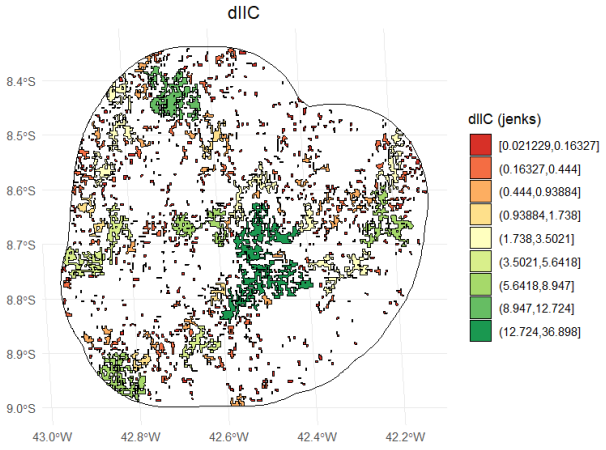
Figura 7. Índice BWC



3.3. IIC y sus fracciones (dIIC, dIICintra, dIICflux, dIICconnector, dIICrest)

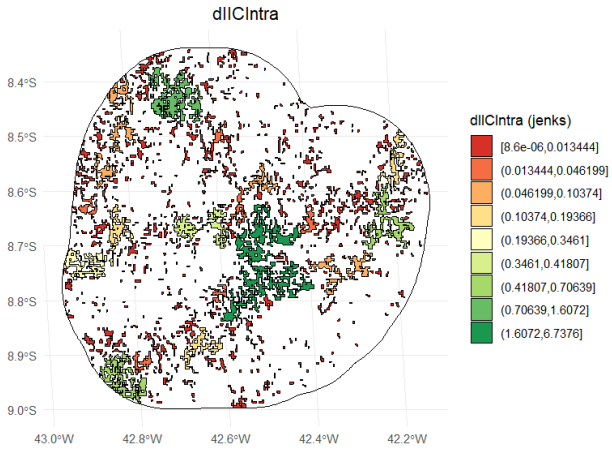
Los parches en verde oscuro tienen los valores más altos (12,724-36,898), indicando que su remoción causaría la mayor pérdida de conectividad, principalmente ubicados en el centro y noroeste. Los fragmentos rojos presentan valores bajos, sugiriendo menor impacto en la conectividad general si fueran removidos, lo que identifica las áreas prioritarias para conservación basándose en su contribución crítica a la conectividad del paisaje (figura 8).

Figura 8. Índice dICC



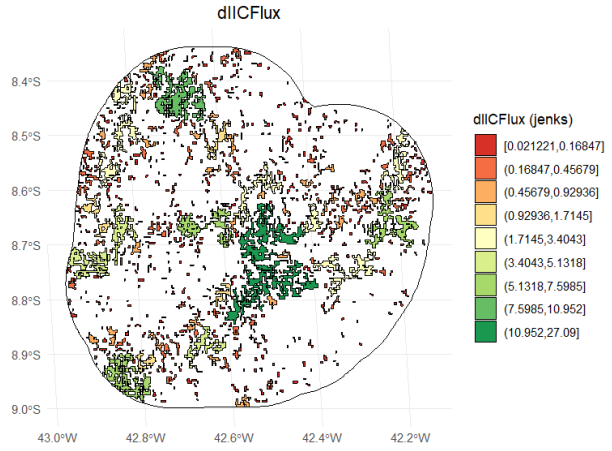
Los parches en verde oscuro tienen los valores más altos (1.6072-6.7376), indicando su mayor contribución a la conectividad interna del hábitat, concentrados principalmente en el centro y noroeste. Los fragmentos rojos presentan valores muy bajos, sugiriendo menor importancia para la conectividad intra-hábitat, lo que ayuda a priorizar áreas críticas para mantener la cohesión funcional dentro de cada tipo de hábitat específico (figura 9).

Figura 9. Índice dIICIntra



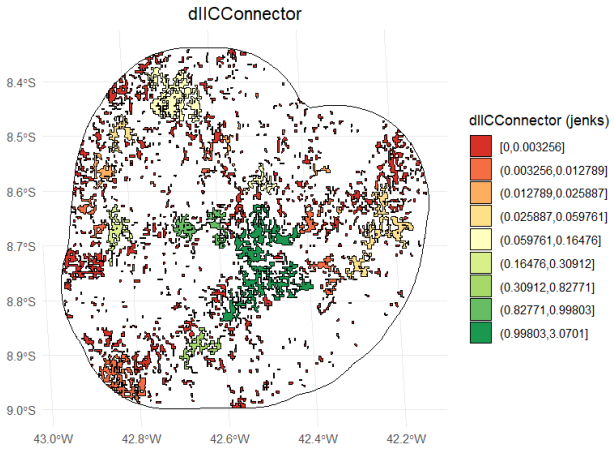
Los parches en verde oscuro tienen los valores más altos (10.952-27.09), indicando su papel crucial como facilitadores del flujo de organismos entre fragmentos, concentrados en el centro y noroeste. Los fragmentos rojos presentan valores bajos, sugiriendo menor capacidad para facilitar la dispersión entre hábitats, lo que identifica las áreas más importantes para mantener la conectividad funcional y los movimientos ecológicos en el paisaje (figura 10).

Figura 10. Índice dIICFlux



Los parches en verde oscuro tienen los valores más altos (0.99603-3.0701), indicando su papel crítico como puentes que facilitan la conectividad entre fragmentos distantes, principalmente ubicados en el centro del paisaje. Los fragmentos rojos presentan valores muy bajos, sugiriendo menor función conectora, lo que identifica las áreas clave que actúan como corredores ecológicos esenciales para mantener la cohesión del paisaje fragmentado (figura 11).

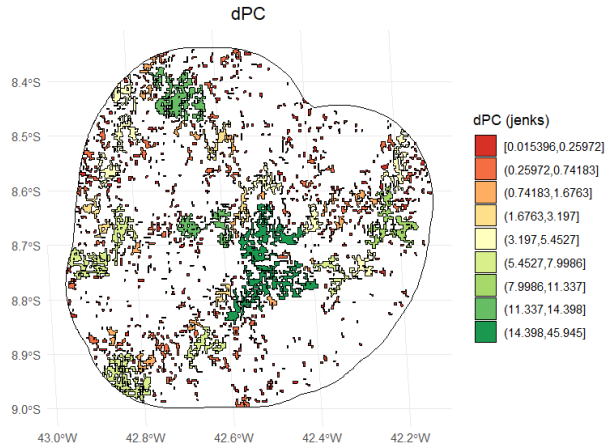
Figura 11. Índice dIICConnector



3.4. PC (Probability of Connectivity) y sus fracciones (dPC, dPCintra, dPCflux, dPCconnector, dPCrest).

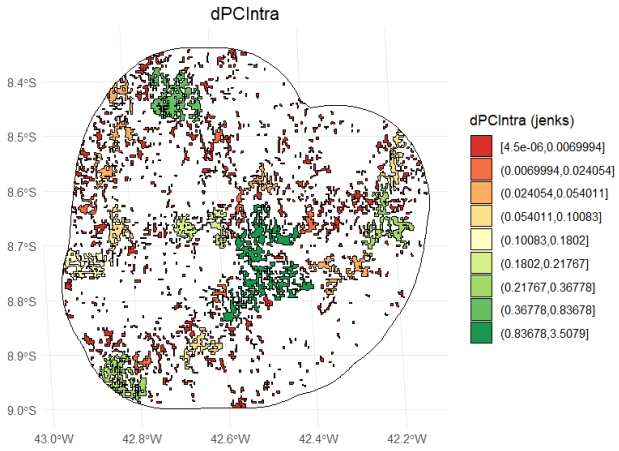
Los parches en verde oscuro tienen los valores más altos (14.398-45.045), indicando que su remoción causaría la mayor reducción en la probabilidad de conectividad, concentrados principalmente en el centro y noroeste del paisaje. Los fragmentos rojos presentan valores bajos, sugiriendo menor impacto en la conectividad probabilística si fueran removidos, lo que identifica las áreas más críticas para preservar la conectividad funcional del sistema (figura 12).

Figura 12. Índice dPC



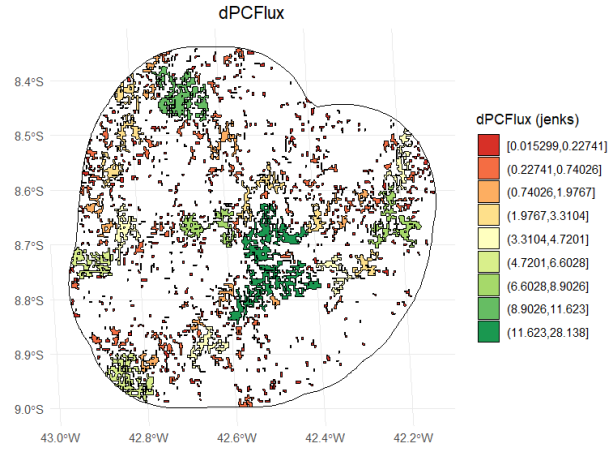
Los parches en verde oscuro tienen los valores más altos (0.83678-3.5079), indicando su mayor contribución a la conectividad probabilística interna del hábitat, concentrados en el centro y noroeste. Los fragmentos rojos presentan valores muy bajos, sugiriendo menor impacto en la conectividad intra-hábitat, lo que identifica las áreas más críticas para preservar la cohesión funcional dentro de cada tipo específico de hábitat (figura 13).

Figura 13. Índice dPCIntra



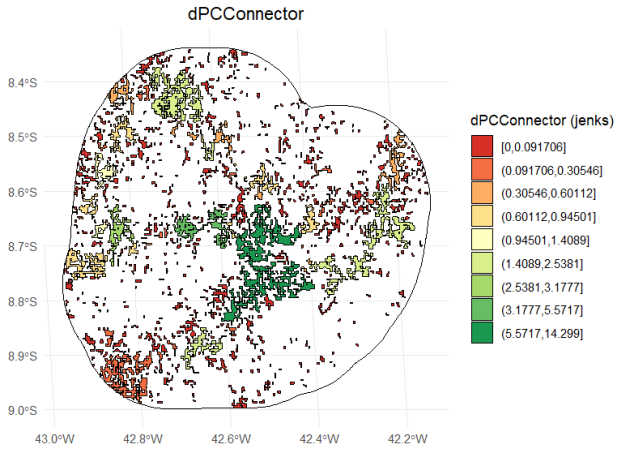
Los parches en verde oscuro tienen los valores más altos (11.623-28.138), indicando su papel crucial como facilitadores de los movimientos probabilísticos de organismos entre fragmentos, concentrados en el centro y noroeste. Los fragmentos rojos presentan valores bajos, sugiriendo menor capacidad para facilitar la dispersión probabilística, identificando las áreas más importantes para mantener los flujos ecológicos en el paisaje (figura 14).

Figura 14. Índice dPCFlux



Los parches en verde oscuro tienen los valores más altos (5.5717-14.299), indicando su papel crítico como puentes que facilitan la conectividad probabilística entre fragmentos distantes, concentrados en el centro del paisaje. Los fragmentos rojos presentan valores muy bajos, sugiriendo menor función conectora probabilística, identificando las áreas clave que actúan como corredores ecológicos esenciales para la conectividad del sistema (figura 15).

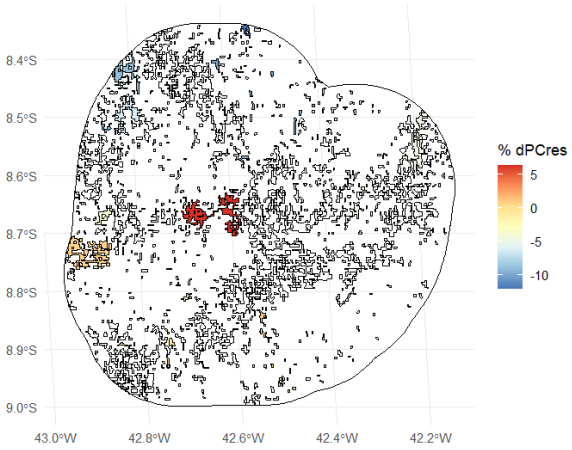
Figura 15. Índice dPCConnector



IICf, PCf y CCIf (Índices focales y compuesto)

Los fragmentos en rojo representan áreas donde la restauración generaría el mayor incremento en conectividad (valores positivos hasta +5%), concentrados principalmente en el centro-este del paisaje. Los parches en azul indican áreas donde la restauración tendría menor impacto o incluso efectos negativos en la conectividad (-10% a valores negativos), identificando las ubicaciones más estratégicas para intervenciones de restauración ecológica basadas en su potencial para mejorar la conectividad del paisaje (figura 16).

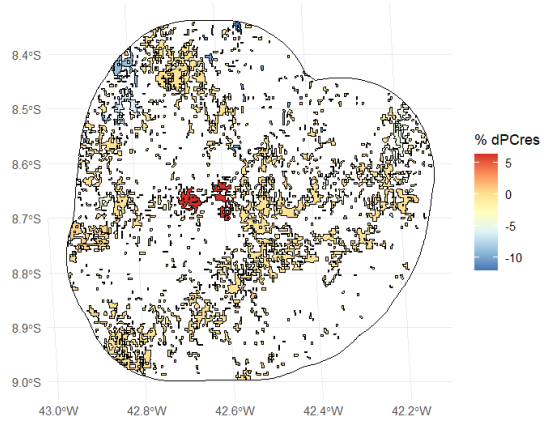
Figura 16. Parches de restauración.



Los fragmentos en rojo representan las ubicaciones más estratégicas para restauración, donde las intervenciones generarían el mayor incremento en conectividad (hasta +5%), concentrados principalmente en el centro-este del paisaje. Los parches en azul indican áreas con menor potencial de mejora o incluso efectos negativos en la conectividad (-10% a valores negativos), identificando

de manera precisa dónde enfocar los esfuerzos de restauración para maximizar los beneficios en la conectividad del paisaje.

Figura 17. Áreas prioritarias de restauración.



Los datos revelan un paisaje altamente fragmentado donde pocos fragmentos centrales (especialmente el fragmento 1) actúan como conectores críticos del sistema, mientras que la mayoría presenta conectividad mínima. La conectividad del paisaje depende principalmente de los flujos entre fragmentos (dPCFlux) más que de la conectividad interna, confirmando la estructura radial observada en los mapas donde un núcleo central mantiene la cohesión del paisaje fragmentado (figura18).

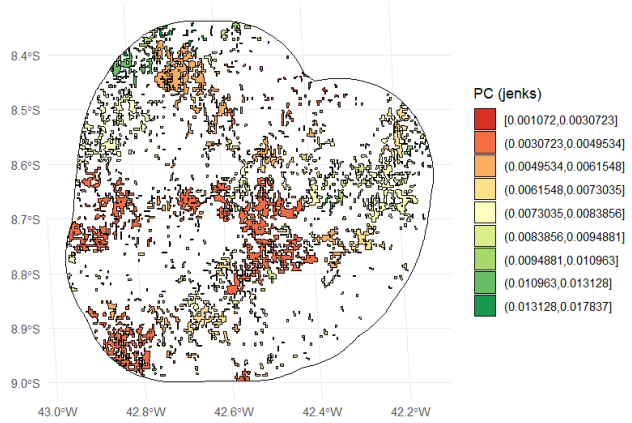
Figura 18. Datos cuantitativos de 683 fragmentos del paisaje analizados mediante múltiples índices de conectividad.

```
Simple feature collection with 683 features and 10 fields
Geometry type: POLYGON
Dimension:     XY
Bounding box:  xmin: -4274640 ymin: -1110576 xmax: -4192392 ymax: -1029925
Projected CRS: World_Mollweide
First 10 features:
   OBJECTID Shape_Leng Shape_Area      area restauracion       dPC  dPCintra   dPCflux
1         2  14167.718  2116605.4 2116605.4            1 0.4264960 0.0010010 0.4254950
2         6   1597.045   141697.4  141697.4            1 0.0261732 0.0000045 0.0261686
3         7   4102.866   814760.2  814760.2            1 0.1075535 0.0001483 0.1074052
4         8  11756.529  2242361.8 2242361.8            0 0.3212590 0.0011235 0.3162666
5         9   1597.045   141697.4  141697.4            1 0.0163085 0.0000045 0.0162197
6        10   9986.199  1753505.7 1753505.7            1 0.2722086 0.0006870 0.2609710
7        12   1597.045   141697.4  141697.4            1 0.0161096 0.0000045 0.0159351
8        15   5388.069   602214.1  602214.1            1 0.0966334 0.0000810 0.0964922
9        16   1597.045   141697.4  141697.4            1 0.0300177 0.0000045 0.0201183
10       17   2505.821   247970.5  247970.5            1 0.0462460 0.0000137 0.0462322
   dPCconnector    dPCres                       geometry
1  1.282308e-14   0.00000 POLYGON ((-4252547 -1030724...
2  1.063710e-07   0.00000 POLYGON ((-4250152 -1031522...
3  1.054712e-14   0.00000 POLYGON ((-4234447 -1031256...
4  3.868963e-03 -12.08704 POLYGON ((-4236577 -1029925...
5  8.430379e-05   0.00000 POLYGON ((-4224865 -1037378...
6  1.055060e-02   0.00000 POLYGON ((-4238174 -1031256...
7  1.699905e-04   0.00000 POLYGON ((-4228858 -1034716...
8  6.020996e-05   0.00000 POLYGON ((-4241900 -1032054...
9  9.894910e-03   0.00000 POLYGON ((-4237109 -1030724...
10 1.975503e-14   0.00000 POLYGON ((-4246159 -1029925...
```

3.6. Índice Integral de Conectividad focal (IICf), Probabilidad de Conectividad focal (PCf), Índice de Conectividad Compuesto (CCIf).

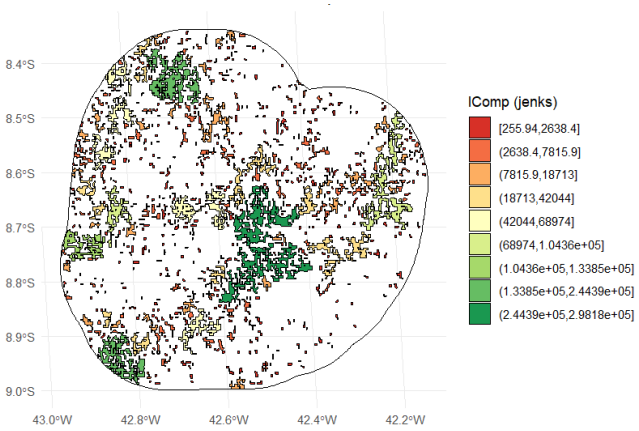
Los parches en verde oscuro presentan los valores más altos de PC (0.013128-0.017837), indicando mayor probabilidad de conectividad y concentrados principalmente en el centro del paisaje. Los fragmentos rojos muestran valores bajos (0.001072-0.0030723), sugiriendo menor probabilidad de conectividad, mientras que la distribución espacial revela un gradiente desde el núcleo central altamente conectado hacia áreas periféricas con conectividad reducida, confirmando la estructura focal del paisaje donde ciertos fragmentos actúan como núcleos de conectividad (figura 19).

Figura 19. Probabilidad de conectividad en paisajes focales.



Los fragmentos en verde oscuro mantienen los valores más altos de conectividad compuesta (2.4439e+05-2.9816e+05), concentrados en el centro y noroeste del paisaje, mientras que los parches rojos presentan valores bajos (255-94.2638.4) en las áreas periféricas. Esta visualización confirma la estructura jerárquica del paisaje donde pocos fragmentos centrales integran múltiples funciones de conectividad, siendo críticos para mantener la cohesión ecológica del sistema fragmentado (figura 20).

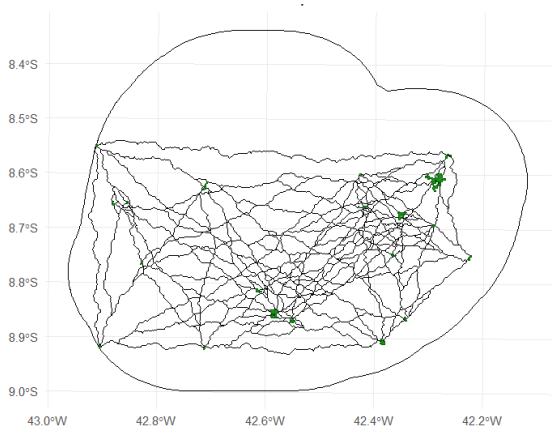
Figura 20. Índice de conectividad compuesta.



3.7. Prioridad de enlaces (eliminación y cambio de enlaces)

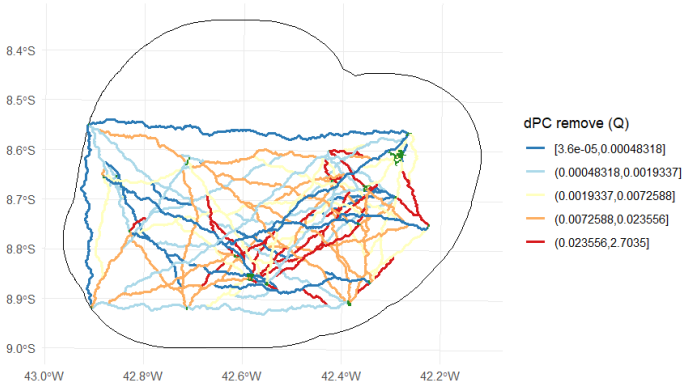
Las líneas grises indican las rutas de conectividad potencial entre parches, mientras que los puntos verdes marcan nodos importantes en la red de conectividad. La visualización revela una estructura de red densa en el centro del paisaje que se extiende hacia áreas periféricas con menor densidad de conexiones. Esta red de corredores identifica las rutas más probables de movimiento de especies entre fragmentos y las áreas críticas donde el establecimiento de corredores ecológicos sería más efectivo para mejorar la conectividad del paisaje fragmentado (figura 21).

Figura 21. Corredores potenciales.



Las líneas están coloreadas según el valor dPC de remoción: las líneas rojas representan enlaces críticos cuya eliminación causaría la mayor pérdida de conectividad (0.023556-2.7035), mientras que las líneas azules indican conexiones menos críticas (3.6e-05 a valores bajos). La red revela que los enlaces más críticos se concentran en el área central del paisaje, identificando las conexiones prioritarias que deben ser protegidas y las rutas de menor importancia donde la pérdida tendría menor impacto en la conectividad general del sistema (figura 22).

Figura 22. Priorización de enlaces.



4. Discusión

La síntesis de los resultados indica que la Caatinga presenta una alta fragmentación (Figura 1) con numerosas islas de vegetación de pequeño tamaño. Según la literatura, la fragmentación y la pérdida de hábitat reducen la biodiversidad y amplifican los efectos a lo largo del tiempo, lo que refuerza la urgencia de medidas de conservación y restauración (Haddad et al., 2015). En nuestro análisis, las medidas de fragmentación (número de fragmentos y tamaño medio) señalan que la mayor parte del área estudiada ha pasado a un estado configuracional que favorece el aislamiento de poblaciones locales (Figura 2,3).

Las métricas de centralidad (Figura 4) identificaron parches con alto grado y betweenness que funcionan como corredores naturales o 'stepping-stones'. Estos nodos deberían ser prioritarios para evitar su pérdida, porque su remoción supone una reducción desproporcionada de la conectividad (Estrada, 2008; Pereira et al., 2017). Los desgloses de IIC y PC (Figuras 8 y 12) permiten distinguir el aporte intrínseco de cada parche (dIICintra/dPCintra) del papel en el flujo regional (dIICflux/dPCflux) y en la conexión mediante puente (dIICconnector/dPCconnector) (Pascual-Hortal y Saura, 2006; Saura y Pascual-Hortal, 2007).

Los índices focales IICf y PCf y el índice compuesto CCIf (Figura 20) localizan zonas donde la acción local (p.ej. restauración dirigida, protección inmediata) tendrá mayor retorno ecológico. La priorización de enlaces (Figura 22) complementa esta visión mostrando qué conexiones, si se restauran o protegen, aumentarían más la conectividad global. Este enfoque combinado (parches + enlaces) es consistente con recomendaciones recientes para planificación de corredores y restauración a escala de paisaje (Hansen et al., 2013; Potapov et al., 2022).

5. Conclusiones

- La Caatinga muestra una fragmentación significativa que pone en riesgo la funcionalidad ecológica del paisaje: alto número de fragmentos y tamaño medio reducido.
- Las métricas de centralidad y las fracciones de IIC/PC identifican parches críticos (nodos y stepping-stones) cuya protección prioritaria debe ser considerada en planes de gestión.
- Los índices focales (IICf, PCf, CCIf) y el análisis de prioridad de enlaces proporcionan una guía espacial concreta para la restauración: enfocar esfuerzos en enlaces y parches que maximicen la ganancia de conectividad.
- Recomendaciones prácticas: conservar parches con alta dIICintra/dPCintra; restaurar enlaces con alta ganancia potencial; integrar medidas socioeconómicas y gobernanza local para asegurar la implementación.
- Limitaciones: los resultados son sensibles a la resolución de los datos y a los supuestos de distancia de dispersión; se recomienda incorporar datos de dispersión de especies locales y validar con observaciones de campo.

6. Bibliografía

Pascual-Hortal, L., & Saura, S. (2006). Comparison and development of new graph-based landscape connectivity indices: Towards the prioritization of habitat patches and corridors for conservation. (Technical report / Conefor).

Saura, S., & Pascual-Hortal, L. (2007). A new habitat availability index to integrate connectivity in landscape conservation planning: Comparison with existing indices and application to a case study. Landscape and Urban Planning, 83(2-3), 91-103. https://doi.org/10.1016/j.landurbplan.2007.03.005

Estrada, E., & Bodin, Ö. (2008). Using network centrality measures to manage landscape connectivity. Ecological Applications, 18(7), 1810-1825. https://doi.org/10.1890/07-1419.1

Haddad, N. M., et al. (2015). Habitat fragmentation and its lasting impact on Earth's ecosystems. Science Advances, 1(2), e1500052. https://doi.org/10.1126/sciadv.1500052

Fahrig, L. (2003). Effects of habitat fragmentation on biodiversity. Annual Review of Ecology, Evolution, and Systematics, 34, 487-515. https://doi.org/10.1146/annurev.ecolsys.34.011802.132419

Hansen, M. C., et al. (2013). High-resolution global maps of 21st-century forest cover change. Science, 342(6160), 850-853. https://doi.org/10.1126/science.1244693

Potapov, P., et al. (2022). Global maps of cropland extent and change show accelerated cropland expansion in the 21st century. Nature Food, 3, 19–28. https://doi.org/10.1038/s43016-021-00429-z

Dryad. Human Footprint Dataset. https://datadryad.org/dataset/doi:10.5061/dryad.ttdz08m1f

SEDAC-NASA. Human Footprint Data Sets. https://www.earthdata.nasa.gov/data/catalog/sedac-ciesin-sedac-lwp3-hf-1993-2018.00

Zenodo. Human Footprint Dataset. https://zenodo.org/records/14449495

Hansen et al. (Global Forest Change dataset). https://earthenginepartners.appspot.com/science-2013-global-forest/download_v1.7.html

GLAD/UMD (Potapov dataset). https://glad.umd.edu/dataset/GLCLUC2020

ESA Land Cover CCI. https://www.esa-landcover-cci.org/

MODIS MCD12Q1. https://lpdaac.usgs.gov/products/mcd12q1v061/

Vieilledent, G., et al. (2022). Forest at Risk. https://forestatrisk.cirad.fr/

GLanCE30 v001 30m. https://lpdaac.usgs.gov/products/glance30v001/

Dynamic World. https://dynamicworld.app/

MapBiomas. https://mapbiomas.org/

CONABIO. http://www.conabio.gob.mx/informacion/gis/

7. Anexo: Script en R utilizado

```
                    ####### PARTE 1#######

library(usethis)

library(devtools)

library(devtools)

library(remotes)

library(igraph)

#install_github("connectscape/Makurhini", dependencies = TRUE, upgrade = "never",force=TRUE)

library(Makurhini)

library(sf)

habitat_nodes <- read_sf("C:/Data_ecologia/Trabajo_final/Final2/Bosque_Caatinga.shp")

nrow(habitat_nodes)

paisaje <- read_sf("C:/Data_ecologia/Trabajo_final/Final2/Paisaje_Estudio.shp")

parches <- read_sf("C:/Data_ecologia/Trabajo_final/Final2/Bosque_Caatinga.shp")

#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 = parches, 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())


library(Makurhini)

library(sf)

vegetation_patches <- sf::read_sf("C:/Data_ecologia/Trabajo_final/Final2/Bosque_Caatinga.shp")

MK_Fragmentation(

  nodes = vegetation_patches,     # Aquí sí pasas el objeto cargado

  edge_distance = 600,

  min_node_area = 100,

  landscape_area = NULL,
```

```
  area_unit = "ha",
  perimeter_unit = "km",
  plot = FALSE,
  write = NULL
)
area_paisaje <- st_area(paisaje)
area_paisaje <- unit_convert(area_paisaje, "m2", "ha")
Fragmentacion <- MK_Fragmentation(nodes = habitat_nodes,
                                  edge_distance = 600,
                                  min_node_area = 100,
                                  landscape_area = area_paisaje,
                                  area_unit = "ha",
                                  perimeter_unit = "km",
                                  plot = TRUE)
ggplot() +
  geom_sf(data = paisaje, aes(color = "Study area"), fill = NA, color = "black") +
  geom_sf(data = Fragmentacion$`Patch statistics shapefile`, aes(fill = CAPercent), color = "black", size = 0.1) +
  scale_fill_distiller(
    palette = "RdYlGn",
    direction = 1,
    name = "% Área Núcleo"
  ) +
  theme_minimal() +
  labs(
    title = "Fragmentación a nivel de parche",
    fill = "% Área Núcleo"
  ) +
  theme(
    legend.position = "right",
    plot.title = element_text(hjust = 0.5)
  )
ggplot() +
  geom_sf(data = paisaje, aes(color = "Study area"), fill = NA, color = "black") +
  geom_sf(data = Fragmentacion$`Patch statistics shapefile`, aes(fill = EdgePercent), color = "black", size = 0.1) +
```

```
  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)
  )
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", size = 0.1) +
  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)
  )
                              ####### PARTE 2#######
# Cargar paquetes necesarios
library(ggplot2)
```

```
library(sf)
library(Makurhini)
library(classInt)
library(dplyr)
library(RColorBrewer)
# Leer tus datos reales
paisaje <- sf::read_sf("C:/Data_ecologia/Trabajo_final/Final2/Paisaje_Estudio.shp")
habitat_nodes <- sf::read_sf("C:/Data_ecologia/Trabajo_final/Final2/Bosque_Caatinga.shp")
# Asegurar que ambos objetos tengan el mismo CRS
habitat_nodes <- st_transform(habitat_nodes, st_crs(paisaje))
# Graficar los parches sobre el paisaje
ggplot() +
  geom_sf(data = paisaje, aes(color = "Área de estudio"), fill = NA, color = "black") +
  geom_sf(data = habitat_nodes, aes(color = "Parches"), fill = "forestgreen", linewidth = 0.5) +
  scale_color_manual(name = "", values = c("Área de estudio" = "black", "Parches" = "forestgreen")) +
  theme_minimal() +
  theme(
    axis.title.x = element_blank(),
    axis.title.y = element_blank(),
    plot.title = element_text(hjust = 0.5)
  ) +
  labs(title = "Distribución de parches en el paisaje")
# Ejecutar análisis de centralidad con tu shapefile de parches
centrality_test <- MK_RMCentrality(
  nodes = habitat_nodes,
  distance = list(type = "centroid"),
  distance_thresholds = 10000,
  probability = 0.5,
  write = NULL
)
# Clasificar la métrica BWC en 5 clases usando Jenks
breaks <- classIntervals(centrality_test$BWC, n = 5, style = "jenks")
centrality_test <- centrality_test %>%
  mutate(BWC_jenks = cut(BWC,
```

```
                    breaks = breaks$brks,
                    include.lowest = TRUE,
                    dig.lab = 6))
# Reproyectar centrality_test si es necesario
centrality_test <- st_transform(centrality_test, st_crs(paisaje))
# Graficar centralidad (BWC) clasificada por Jenks
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)
  )
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)
```

```
  )
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)
  )
centrality_test <- MK_RMCentrality(nodes = habitat_nodes,
                                   distance = list(type = "centroid"),
                                   distance_thresholds = c(10000, 100000),
                                   probability = 0.5,
                                   write = NULL)
centrality_test
plot(centrality_test$d10000["BWC"], breaks = "quantile")
plot(centrality_test$`d1e+05`["BWC"], breaks = "quantile")
                                    ####### PARTE 3#######
library(Makurhini)
library(sf)
habitat_nodes <- read_sf("C:/Data_ecologia/Trabajo_final/Final2/Bosque_Caatinga.shp")
nrow(habitat_nodes)
paisaje <- read_sf("C:/Data_ecologia/Trabajo_final/Final2/Paisaje_Estudio.shp")
parches <- read_sf("C:/Data_ecologia/Trabajo_final/Final2/Bosque_Caatinga.shp")
# 2. Verificar los sistemas de coordenadas
st_crs(paisaje)   # CRS del paisaje
```

```
st_crs(parches)   # CRS de los parches
# 3. Unificar el CRS si son distintos (usaremos WGS84 EPSG:4326)
if (st_crs(paisaje) != st_crs(parches)) {
  parches <- st_transform(parches, crs = st_crs(paisaje))
}
library(ggplot2)
library(sf)
library(Makurhini)
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 = 0.5) +
  scale_color_manual(name = "", values = "black")+
  theme_minimal() +
  theme(axis.title.x = element_blank(),
        axis.title.y = element_blank())
IIC <- MK_dPCIIC(nodes = habitat_nodes,
                 attribute = NULL,
                 area_unit = "ha",
                 distance = list(type = "centroid"),
                 LA = NULL,
                 onlyoverall = TRUE,
                 metric = "IIC",
                 distance_thresholds = 10000,
                 intern = TRUE) #10 km
#> Estimating IIC index. This may take several minutes depending on the number of nodes
#>
|
  |                                      |   0%
#>
#> Done!
IIC
#>     Index       Value
#> 1  IICnum 1.434310e+12
```

```
#> 2 EC(IIC) 1.197627e+06
area_paisaje <- st_area(paisaje)
area_paisaje <- unit_convert(area_paisaje, "m2", "ha")
IIC <- MK_dPCIIC(nodes = habitat_nodes,
                 attribute = NULL,
                 area_unit = "ha",
                 distance = list(type = "centroid"),
                 LA = area_paisaje,
                 onlyoverall = TRUE,
                 metric = "IIC",
                 distance_thresholds = 10000,
                 intern = TRUE) #10 km
#> Estimating IIC index. This may take several minutes depending on the number of nodes
#>
|
  |                                      |   0%
#>
#> Done!
IIC
#>     Index        Value
#> 1  IICnum 1.434310e+12
#> 2 EC(IIC) 1.197627e+06
#> 3     IIC 1.881017e-01
area_paisaje <- st_area(paisaje)
area_paisaje <- unit_convert(area_paisaje, "m2", "ha")
IIC <- MK_dPCIIC(nodes = habitat_nodes,
                 attribute = NULL,
                 area_unit = "ha",
                 distance = list(type = "edge"),
                 LA = area_paisaje,
                 onlyoverall = TRUE,
                 metric = "IIC",
                 distance_thresholds = 10000,
                 intern = TRUE) #10 km
```

```
#> Estimating IIC index. This may take several minutes depending on the number of nodes
#>
|
  |                                      |   0%
#>
#> Done!
IIC
#>     Index        Value
#> 1  IICnum 5.700691e+11
#> 2 EC(IIC) 7.550292e+05
#> 3     IIC 7.476136e-02
#>
#>
area_paisaje <- st_area(paisaje)
area_paisaje <- unit_convert(area_paisaje, "m2", "ha")
IIC <- MK_dPCIIC(nodes = habitat_nodes,
                 attribute = NULL,
                 area_unit = "ha",
                 distance = list(type = "edge", keep = 0.1),
                 LA = area_paisaje,
                 onlyoverall = TRUE,
                 metric = "IIC",
                 distance_thresholds = 10000,
                 intern = TRUE) #10 km
#> Estimating IIC index. This may take several minutes depending on the number of nodes
#>
|
  |                                      |   0%
#>
#> Done!
IIC
#>     Index        Value
#> 1  IICnum 5.693868e+11
#> 2 EC(IIC) 7.545772e+05
```

```
#> 3     IIC 7.467188e-02
area_paisaje <- st_area(paisaje)
area_paisaje <- unit_convert(area_paisaje, "m2", "ha")
IIC <- MK_dPCIIC(nodes = habitat_nodes,
                 attribute = NULL,
                 area_unit = "ha",
                 distance = list(type = "edge", keep = 0.1),
                 LA = area_paisaje,
                 onlyoverall = FALSE,
                 metric = "IIC",
                 distance_thresholds = 10000,
                 intern = TRUE) #10 km
#> Estimating IIC index. This may take several minutes depending on the number of nodes
#>
|
  |                                       |   0%
#>
#> Done!
IIC
library(classInt)
library(igraph)
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(IIC$dIIC, n = 9, style = "jenks")


# Crear una nueva variable categórica con los intervalos
IIC <- IIC %>%
  mutate(dIIC_q = cut(dIIC,
                      breaks = breaks$brks,
                      include.lowest = TRUE,
                      dig.lab = 5))
# Graficar en ggplot2 usando las clases Jenks
ggplot() +
  geom_sf(data = paisaje, fill = NA, color = "black") +
  geom_sf(data = IIC, aes(fill = dIIC_q), color = "black", size = 0.1) +
  scale_fill_brewer(palette = "RdYlGn", direction = 1, name = "dIIC (jenks)") +
  theme_minimal() +
  labs(
    title = "dIIC",
    fill = "dIIC"
  ) +
  theme(
    legend.position = "right",
    plot.title = element_text(hjust = 0.5)
  )
# Calcular los intervalos de Jenks para strength
breaks <- classInt::classIntervals(IIC$dIICintra, n = 9, style = "jenks")
# Crear una nueva variable categórica con los intervalos
IIC <- IIC %>%
  mutate(dIIC_q = cut(dIICintra,
                      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 = IIC, aes(fill = dIIC_q), color = "black", size = 0.1) +
```

```
  scale_fill_brewer(palette = "RdYlGn", direction = 1, name = "dIICIntra (jenks)") +
  theme_minimal() +
  labs(
    title = "dIICIntra",
    fill = "dIICIntra"
  ) +
  theme(
    legend.position = "right",
    plot.title = element_text(hjust = 0.5)
  )
# Calcular los intervalos de Jenks para strength
breaks <- classInt::classIntervals(IIC$dIICflux, n = 9, style = "jenks")

# Crear una nueva variable categórica con los intervalos
IIC <- IIC %>%
  mutate(dIIC_q = cut(dIICflux,
                      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 = IIC, aes(fill = dIIC_q), color = "black", size = 0.1) +
  scale_fill_brewer(palette = "RdYlGn", direction = 1, name = "dIICFlux (jenks)") +
  theme_minimal() +
  labs(
    title = "dIICFlux",
    fill = "dIICFlux"
  ) +
  theme(
    legend.position = "right",
    plot.title = element_text(hjust = 0.5)
  )
# Calcular los intervalos de Jenks para strength
```

```
breaks <- classInt::classIntervals(IIC$dIICconnector, n = 9, style = "jenks")
# Crear una nueva variable categórica con los intervalos
IIC <- IIC %>%
  mutate(dIIC_q = cut(dIICconnector,
                      breaks = breaks$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 = IIC, aes(fill = dIIC_q), color = "black", size = 0.1) +
  scale_fill_brewer(palette = "RdYlGn", direction = 1, name = "dIICConnector (jenks)") +
  theme_minimal() +
  labs(
    title = "dIICConnector",
    fill = "dIICConnector"
  ) +
  theme(
    legend.position = "right",
    plot.title = element_text(hjust = 0.5)
  )
# Calcular el área en m²
area_paisaje <- st_area(paisaje)

# Convertir el área a hectáreas (1 ha = 10,000 m²)
area_paisaje <- as.numeric(area_paisaje) / 10000

# Calcular el índice IIC para distintos umbrales de distancia
IIC <- MK_dPCIIC(
  nodes = habitat_nodes,
  attribute = NULL,
  area_unit = "ha",
  distance = list(type = "edge", keep = 0.1),
  LA = area_paisaje,
```

```
  overall = TRUE,
  onlyoverall = FALSE,
  metric = "IIC",
  distance_thresholds = c(2000, 10000, 50000),
  intern = TRUE
)
# Mostrar resultados
print(IIC)
PC <- MK_dPCIIC(nodes = habitat_nodes,
                attribute = NULL,
                area_unit = "ha",
                distance = list(type = "edge", keep = 0.1),
                LA = NULL,
                onlyoverall = TRUE,
                metric = "PC",
                probability = 0.5,
                distance_thresholds = 10000,
                intern = TRUE) #10 km
# Mostrar resultados
print(IIC)


PC <- MK_dPCIIC(nodes = habitat_nodes,
                attribute = NULL,
                area_unit = "ha",
                distance = list(type = "centroid"),
                LA = area_paisaje,
                onlyoverall = TRUE,
                metric = "PC",
                probability = 0.5,
                distance_thresholds = 10000,
                intern = TRUE) #10 km
area_paisaje <- st_area(paisaje)
area_paisaje <- unit_convert(area_paisaje, "m2", "ha")
```

```
PC <- MK_dPCIIC(nodes = habitat_nodes,
                attribute = NULL,
                area_unit = "ha",
                distance = list(type = "edge", keep = 0.1),
                LA = area_paisaje,
                onlyoverall = FALSE,
                metric = "PC",
                probability = 0.5,
                distance_thresholds = 10000,
                intern = TRUE) #10 km
# Mostrar resultados
print(IIC)
                        #############PARTE 4#########################
library(classInt)
library(dplyr)
# Calcular los intervalos de Jenks para strength
breaks <- classInt::classIntervals(PC$dPC, n = 9, style = "jenks")
# Crear una nueva variable categórica con los intervalos
PC <- PC %>%
  mutate(dPC_q = cut(dPC,
                     breaks = breaks$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 = PC, aes(fill = dPC_q), color = "black", size = 0.1) +
  scale_fill_brewer(palette = "RdYlGn", direction = 1, name = "dPC (jenks)") +
  theme_minimal() +
  labs(
    title = "dPC",
    fill = "dPC"
  ) +
  theme(
```

```
    legend.position = "right",
    plot.title = element_text(hjust = 0.5)
  )
# Calcular los intervalos de Jenks para strength
breaks <- classInt::classIntervals(PC$dPCintra, n = 9, style = "jenks")
# Crear una nueva variable categórica con los intervalos
PC <- PC %>%
  mutate(dPC_q = cut(dPCintra,
                     breaks = breaks$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 = PC, aes(fill = dPC_q), color = "black", size = 0.1) +
  scale_fill_brewer(palette = "RdYlGn", direction = 1, name = "dPCIntra (jenks)") +
  theme_minimal() +
  labs(
    title = "dPCIntra",
    fill = "dPCIntra"
  ) +
  theme(
    legend.position = "right",
    plot.title = element_text(hjust = 0.5)
  )
# Calcular los intervalos de Jenks para strength
breaks <- classInt::classIntervals(PC$dPCflux, n = 9, style = "jenks")
# Crear una nueva variable categórica con los intervalos
PC <- PC %>%
  mutate(dPC_q = cut(dPCflux,
                     breaks = breaks$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 = PC, aes(fill = dPC_q), color = "black", size = 0.1) +
  scale_fill_brewer(palette = "RdYlGn", direction = 1, name = "dPCFlux (jenks)") +
  theme_minimal() +
  labs(
    title = "dPCFlux",
    fill = "dPCFlux"
  ) +
  theme(
    legend.position = "right",
    plot.title = element_text(hjust = 0.5)
  )
# Calcular los intervalos de Jenks para strength
breaks <- classInt::classIntervals(PC$dPCconnector, n = 9, style = "jenks")
# Crear una nueva variable categórica con los intervalos
PC <- PC %>%
  mutate(dPC_q = cut(dPCconnector,
                     breaks = breaks$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 = PC, aes(fill = dPC_q), color = "black", size = 0.1) +
  scale_fill_brewer(palette = "RdYlGn", direction = 1, name = "dPCConnector (jenks)") +
  theme_minimal() +
  labs(
    title = "dPCConnector",
    fill = "dPCConnector"
  ) +
  theme(
    legend.position = "right",
```

```
    plot.title = element_text(hjust = 0.5)
  )
area_paisaje <- st_area(paisaje)
area_paisaje <- unit_convert(area_paisaje, "m2", "ha")
PC <- MK_dPCIIC(nodes = habitat_nodes,
                attribute = NULL,
                area_unit = "ha",
                distance = list(type = "edge", keep = 0.1),
                LA = area_paisaje,
                overall = TRUE,
                onlyoverall = FALSE,
                metric = "PC",
                probability = 0.5,
                distance_thresholds = 10000,
                intern = TRUE) #10 km
library(usethis)
library(devtools)
library(devtools)
library(remotes)
library(igraph)
#install_github("connectscape/Makurhini", dependencies = TRUE, upgrade = "never",force=TRUE)
library(Makurhini)
library(sf)
habitat_nodes <- read_sf("C:/Data_ecologia/Trabajo_final/Final2/Bosque_Caatinga.shp")
nrow(habitat_nodes)
paisaje <- read_sf("C:/Data_ecologia/Trabajo_final/Final2/Paisaje_Estudio.shp")
parches <- read_sf("C:/Data_ecologia/Trabajo_final/Final2/Bosque_Caatinga.shp")
#install.packages("ggplot2", dependencies = TRUE)
#install.packages("RColorBrewer"), dependencies = TRUE)
library(ggplot2)
library(RColorBrewer)
library(igraph)
library(terra)
#> terra 1.8.29
```

```
#>
#> Adjuntando el paquete: 'terra'
#> The following objects are masked from 'package:igraph':
#>
#>     blocks, compare
data("resistance_matrix", package = "Makurhini")
raster_map <- as(resistance_matrix, "SpatialPixelsDataFrame")
raster_map <- as.data.frame(raster_map)
colnames(raster_map) <- c("value", "x", "y")
ggplot() +
  geom_tile(data = raster_map, aes(x = x, y = y, fill = value), alpha = 0.8) +
  geom_sf(data = paisaje, aes(color = "Study area"), fill = NA, color = "black") +
  geom_sf(data = habitat_nodes, aes(color = "Habitat nodes"), fill = "forestgreen", linewidth = 0.5) +
  scale_fill_gradientn(colors = c("#000004FF", "#1B0C42FF", "#4B0C6BFF", "#781C6DFF",
                                  "#A52C60FF", "#CF4446FF", "#ED6925FF", "#FB9A06FF",
                                  "#F7D03CFF", "#FCFFA4FF"))+
  scale_color_manual(name = "", values = "black")+
  theme_minimal() +
  theme(axis.title.x = element_blank(),
        axis.title.y = element_blank())
library(usethis)
library(devtools)
library(remotes)
library(igraph)
library(Makurhini)
library(sf)
library(ggplot2)
library(RColorBrewer)
library(terra)
# Cargar tus datos
habitat_nodes <- read_sf("C:/Data_ecologia/Trabajo_final/Final2/Bosque_Caatinga.shp")
paisaje <- read_sf("C:/Data_ecologia/Trabajo_final/Final2/Paisaje_Estudio.shp")
# Verificar sistemas de coordenadas
print("CRS del paisaje:")
```

```
print(st_crs(paisaje))
print("CRS de habitat_nodes:")
print(st_crs(habitat_nodes))
# Asegurar que ambos tengan el mismo CRS
if(st_crs(paisaje) != st_crs(habitat_nodes)) {
  habitat_nodes <- st_transform(habitat_nodes, st_crs(paisaje))
}
# OPCIÓN 1: Mapa simple sin raster de fondo
ggplot() +
  geom_sf(data = paisaje, fill = "lightgray", color = "black", linewidth = 1) +
  geom_sf(data = habitat_nodes, fill = "forestgreen", color = "darkgreen", size = 2) +
  theme_minimal() +
  theme(axis.title.x = element_blank(),
        axis.title.y = element_blank()) +
  labs(title = "Distribución de parches de bosque Caatinga",
       subtitle = "Área de estudio y nodos de hábitat")
# OPCIÓN 2: Si tienes una columna con valores de conectividad en habitat_nodes
# Reemplaza "tu_columna_valor" con el nombre real de tu columna
if("area" %in% colnames(habitat_nodes) | "connectivity" %in% colnames(habitat_nodes)) {
  # Usar la columna que tengas (área o conectividad)
  valor_columna <- ifelse("connectivity" %in% colnames(habitat_nodes), "connectivity", "area")
  ggplot() +
    geom_sf(data = paisaje, fill = "lightgray", color = "black", linewidth = 1) +
    geom_sf(data = habitat_nodes, aes(fill = get(valor_columna)),
            color = "darkgreen", size = 2) +
    scale_fill_viridis_c(name = paste("Valor de\n", valor_columna),
                         option = "plasma") +
    theme_minimal() +
    theme(axis.title.x = element_blank(),
          axis.title.y = element_blank()) +
    labs(title = "Conectividad de parches de bosque Caatinga",
         subtitle = paste("Coloreado por", valor_columna))
}
```

```
# OPCIÓN 3: Crear un raster de resistencia basado en tus datos
# Solo si quieres un fondo de resistencia
# Obtener la extensión de tu área de estudio
bbox <- st_bbox(paisaje)
# Crear un grid regular dentro de tu área de estudio
grid_resolution <- 1000  # metros, ajusta según necesites
grid <- st_make_grid(paisaje,
                     cellsize = grid_resolution,
                     what = "centers") %>%
  st_sf() %>%
  st_filter(paisaje)  # Solo puntos dentro del área de estudio
# Calcular distancia a parches más cercanos para crear resistencia
grid$resistance <- as.numeric(st_distance(grid, habitat_nodes, by_element = FALSE)[,1])
# Normalizar valores de resistencia (0-100)
grid$resistance_norm <- scales::rescale(grid$resistance, to = c(0, 100))
# Extraer coordenadas para ggplot
coords <- st_coordinates(grid)
grid_df <- data.frame(
  x = coords[,1],
  y = coords[,2],
  resistance = grid$resistance_norm
)
# Mapa final con raster de resistencia basado en tus datos
ggplot() +
  geom_point(data = grid_df, aes(x = x, y = y, color = resistance),
             size = 0.5, alpha = 0.7) +
  geom_sf(data = paisaje, fill = NA, color = "black", linewidth = 1.5) +
  geom_sf(data = habitat_nodes, fill = "forestgreen", color = "darkgreen", size = 3) +
  scale_color_gradientn(name = "Resistencia\n(distancia a\nbosque)",
                        colors = c("#000004FF", "#1B0C42FF", "#4B0C6BFF", "#781C6DFF",
                                   "#A52C60FF", "#CF4446FF", "#ED6925FF", "#FB9A06FF",
                                   "#F7D03CFF", "#FCFFA4FF")) +
  theme_minimal() +
  theme(axis.title.x = element_blank(),
```

```
      axis.title.y = element_blank()) +
  labs(title = "Conectividad del paisaje - Bosque Caatinga",
       subtitle = "Resistencia basada en distancia a parches de bosque")
# Verificar la extensión de tus datos
print("Extensión del paisaje:")
print(st_bbox(paisaje))
print("Extensión de habitat_nodes:")
print(st_bbox(habitat_nodes))
# ===== CREAR MATRIZ DE RESISTENCIA PARA TUS DATOS =====
# Opción 1: Crear raster de resistencia simple basado en distancia euclidiana
library(terra)
# Crear un raster que cubra tu área de estudio
bbox <- st_bbox(paisaje)
resolution <- 100  # 100 metros de resolución, ajusta según necesites
# Crear raster vacío
raster_template <- rast(xmin = bbox[1], xmax = bbox[3],
                        ymin = bbox[2], ymax = bbox[4],
                        resolution = resolution,
                        crs = st_crs(paisaje)$wkt)
# Asignar valores de resistencia
# Valor 1 para bosque (baja resistencia), valor 100 para no-bosque (alta resistencia)
resistance_raster <- rasterize(vect(habitat_nodes), raster_template, field = 1)
resistance_raster[is.na(resistance_raster)] <- 100  # Áreas sin bosque = alta resistencia
# Convertir a formato que acepta Makurhini
resistance_matrix_custom <- resistance_raster
# Ahora ejecutar el análisis de conectividad con TUS datos
PC <- MK_dPCIIC(nodes = habitat_nodes,
                attribute = NULL,  # o especifica columna de área si la tienes
                distance = list(type = "least-cost",
                                resistance = resistance_matrix_custom),
                metric = "PC",
                probability = 0.5,
                overall = FALSE,
                distance_thresholds = 10000) # 10 km
```

```
# Ver resultados
print("Resultados de conectividad PC:")
print(head(PC$dPCIIC))
# ALTERNATIVA: Si quieres usar distancia euclidiana (más simple)
PC_euclidean <- MK_dPCIIC(nodes = habitat_nodes,
                          attribute = NULL,
                          distance = list(type = "centroid"),  # distancia euclidiana
                          metric = "PC",
                          probability = 0.5,
                          overall = FALSE,
                          distance_thresholds = 10000)
print("Resultados con distancia euclidiana:")
print(head(PC_euclidean$dPCIIC))
# Visualizar resultados de conectividad
if(!is.null(PC$dPCIIC)) {
  # Agregar valores de PC a los nodos
  habitat_nodes$PC_value <- PC$dPCIIC$dPC
  # Mapa con valores de conectividad
  ggplot() +
    geom_sf(data = paisaje, fill = "lightgray", color = "black", linewidth = 1) +
    geom_sf(data = habitat_nodes, aes(fill = PC_value),
            color = "darkgreen", size = 3) +
    scale_fill_viridis_c(name = "Valor PC\n(Conectividad)",
                         option = "plasma") +
    theme_minimal() +
    theme(axis.title.x = element_blank(),
          axis.title.y = element_blank(),
          legend.position = "right") +
    labs(title = "Índice de Conectividad PC - Bosque Caatinga",
         subtitle = "Valores dPC para cada parche de bosque")
}
```

```
library(classInt)
library(igraph)
library(dplyr)
# Calcular los intervalos de Jenks para strength
breaks <- classInt::classIntervals(PC$dPC, n = 9, style = "jenks")
# Crear una nueva variable categórica con los intervalos
PC <- PC %>%
  mutate(dPC_q = cut(dPC,
                     breaks = breaks$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 = PC, aes(fill = dPC_q), color = "black", size = 0.1) +
  scale_fill_brewer(palette = "RdYlGn", direction = 1, name = "dPC (jenks)") +
  theme_minimal() +
  labs(
    title = "dPC",
    fill = "dPC"
  ) +
  theme(
    legend.position = "right",
    plot.title = element_text(hjust = 0.5)
  )
# Calcular los intervalos de Jenks para strength
breaks <- classInt::classIntervals(PC$dPCintra, n = 9, style = "jenks")
# Crear una nueva variable categórica con los intervalos
PC <- PC %>%
  mutate(dPC_q = cut(dPCintra,
                     breaks = breaks$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 = PC, aes(fill = dPC_q), color = "black", size = 0.1) +
  scale_fill_brewer(palette = "RdYlGn", direction = 1, name = "dPCIntra (jenks)") +
  theme_minimal() +
  labs(
    title = "dPCIntra",
    fill = "dPCIntra"
  ) +
  theme(
    legend.position = "right",
    plot.title = element_text(hjust = 0.5)
  )
# Calcular los intervalos de Jenks para strength
breaks <- classInt::classIntervals(PC$dPCflux, n = 9, style = "jenks")
# Crear una nueva variable categórica con los intervalos
PC <- PC %>%
  mutate(dPC_q = cut(dPCflux,
                     breaks = breaks$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 = PC, aes(fill = dPC_q), color = "black", size = 0.1) +
  scale_fill_brewer(palette = "RdYlGn", direction = 1, name = "dPCFlux (jenks)") +
  theme_minimal() +
  labs(
    title = "dPCFlux",
    fill = "dPCFlux"
  ) +
  theme(
    legend.position = "right",
```

```
    plot.title = element_text(hjust = 0.5)
  )
# Calcular los intervalos de Jenks para strength
breaks <- classInt::classIntervals(PC$dPCconnector, n = 9, style = "jenks")
# Crear una nueva variable categórica con los intervalos
PC <- PC %>%
  mutate(dPC_q = cut(dPCconnector,
                     breaks = breaks$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 = PC, aes(fill = dPC_q), color = "black", size = 0.1) +
  scale_fill_brewer(palette = "RdYlGn", direction = 1, name = "dPCConnector (jenks)") +
  theme_minimal() +
  labs(
    title = "dPCConnector",
    fill = "dPCConnector"
  ) +
  theme(
    legend.position = "right",
    plot.title = element_text(hjust = 0.5)
  )
################ PARTE 5#####################################
library(ggplot2)
library(sf)
library(terra)
library(raster)
library(Makurhini)
library(RColorBrewer)
# Cargar tus datos
habitat_nodes <- read_sf("C:/Data_ecologia/Trabajo_final/Final2/Bosque_Caatinga.shp")
paisaje <- read_sf("C:/Data_ecologia/Trabajo_final/Final2/Paisaje_Estudio.shp")
```

```
# Verificar sistemas de coordenadas
print("CRS del paisaje:")
print(st_crs(paisaje))
print("CRS de habitat_nodes:")
print(st_crs(habitat_nodes))
# Asegurar que ambos tengan el mismo CRS
if(st_crs(paisaje) != st_crs(habitat_nodes)) {
  habitat_nodes <- st_transform(habitat_nodes, st_crs(paisaje))
}
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())
set.seed(10) #Para que seleccionen los mismos parches que yo
#Seleccionar de forma aleatoria a 100 de estos parches para restaurar
parches_restauracion <- sample(1:nrow(habitat_nodes), 100)
parches_restauracion
habitat_nodes$restauracion <- 1
habitat_nodes$restauracion[parches_restauracion] <- 0
PCrestauracion <- MK_dPCIIC(nodes = habitat_nodes,
                            attribute = NULL,
                            area_unit = "ha",
                            distance = list(type = "edge", keep = 0.1),
                            LA = NULL,
                            restoration = "restauracion",
                            onlyrestor = TRUE,
                            metric = "PC",
                            probability = 0.5,
                            distance_thresholds = 10000,
                            intern = TRUE) #10 km
```

```
PCrestauracion
ggplot() +
  geom_sf(data = paisaje, aes(color = "Study area"), fill = NA, color = "black") +
  geom_sf(data = PCrestauracion, aes(fill = dPCres), color = "black", size = 0.1) +
  scale_fill_distiller(
    palette = "RdYlBu",
    direction = -1,
    name = "% dPCres"
  ) +
  theme_minimal() +
  labs(
    title = "Restauración",
    fill = "% dPCres"
  ) +
  theme(
    legend.position = "right",
    plot.title = element_text(hjust = 0.5)
  )
PCrestauracion2 <- PCrestauracion[PCrestauracion$restauracion == 0,]
ggplot() +
  geom_sf(data = paisaje, aes(color = "Study area"), fill = NA, color = "black") +
  geom_sf(data = habitat_nodes, aes(color = "Patches"), fill = NA, color = "black") +
  geom_sf(data = PCrestauracion2, aes(fill = dPCres), color = "black", size = 0.1) +
  scale_fill_distiller(
    palette = "RdYlBu",
    direction = -1,
    name = "% dPCres"
  ) +
  theme_minimal() +
  labs(
    title = "Restauración",
    fill = "% dPCres"
  ) +
  theme(
```

```
  legend.position = "right",
  plot.title = element_text(hjust = 0.5)
 )
PCrestauracion <- MK_dPCIIC(nodes = habitat_nodes,
                            attribute = NULL,
                            area_unit = "ha",
                            distance = list(type = "edge", keep = 0.1),
                            LA = NULL,
                            restoration = "restauracion",
                            onlyrestor = FALSE,
                            metric = "PC",
                            probability = 0.5,
                            distance_thresholds = 10000,
                            intern = FALSE) #10 km
PCrestauracion
##############Parte 6###############
library(Makurhini)
library(sf)
library(ggplot2)
library(RColorBrewer)
library(classInt)
library(dplyr)
# Cargar datos
habitat_nodes <- read_sf("C:/Data_ecologia/Trabajo_final/Final2/Bosque_Caatinga.shp")
paisaje <- read_sf("C:/Data_ecologia/Trabajo_final/Final2/Paisaje_Estudio.shp")
parches <- read_sf("C:/Data_ecologia/Trabajo_final/Final2/Bosque_Caatinga.shp")
# Crear ID si no existe
if(!"Id" %in% names(habitat_nodes)) {
  habitat_nodes$Id <- 1:nrow(habitat_nodes)
  print("Columna Id creada")
}
# Muestra de 10 parches
set.seed(123)
sample_10 <- habitat_nodes[sample(1:nrow(habitat_nodes), 10), ]
```

```
# Análisis de conectividad
start_time <- Sys.time()
cat("Iniciando análisis con 10 parches...\n")
test_10 <- MK_Focal_nodes(nodes = sample_10,
                          id = "Id",
                          attribute = NULL,
                          raster_attribute = NULL,
                          fun_attribute = NULL,
                          distance = list(type = "edge", keep = 0.1),
                          metric = "PC",
                          probability = 0.5,
                          distance_thresholds = 10000,
                          search_buffer = 20000,
                          fragmentation = FALSE,
                          parallel = 4,
                          intern = FALSE)
end_time <- Sys.time()
cat("Tiempo:", round(as.numeric(difftime(end_time, start_time, units = "mins")), 2), "minutos\n")
# CORRECCIÓN: Usar test_10 en lugar de test
breaks <- classInt::classIntervals(test_10$PC, n = 9, style = "jenks")
PC <- test_10 %>%
  mutate(dPC_q = cut(PC,
                     breaks = breaks$brks,
                     include.lowest = TRUE,
                     dig.lab = 5))
# Visualización
ggplot() +
  geom_sf(data = paisaje, fill = NA, color = "black") +
  geom_sf(data = PC, aes(fill = dPC_q), color = "black", size = 0.1) +
  scale_fill_brewer(palette = "RdYlGn", direction = 1, name = "PC (jenks)") +
  theme_minimal() +
  labs(
    title = "PC en paisajes focales (muestra 10 parches)",
```

```
    fill = "PC"
  ) +
  theme(
    legend.position = "right",
    plot.title = element_text(hjust = 0.5)
  )
###################################################################### para los 863 parches
library(Makurhini)
library(sf)
library(ggplot2)
library(RColorBrewer)
library(classInt)
library(dplyr)
# Cargar datos
habitat_nodes <- read_sf("C:/Data_ecologia/Trabajo_final/Final2/Bosque_Caatinga.shp")
paisaje <- read_sf("C:/Data_ecologia/Trabajo_final/Final2/Paisaje_Estudio.shp")
parches <- read_sf("C:/Data_ecologia/Trabajo_final/Final2/Bosque_Caatinga.shp")
# Crear ID si no existe
if(!"Id" %in% names(habitat_nodes)) {
  habitat_nodes$Id <- 1:nrow(habitat_nodes)
  print("Columna Id creada")
}
# Verificar IDs únicos
if(anyDuplicated(habitat_nodes$Id)) {
  habitat_nodes$Id <- 1:nrow(habitat_nodes)
  print("IDs duplicados corregidos")
}
# Guardar backup antes del análisis
save.image("backup_antes_analisis_863_parches.RData")
# ANÁLISIS COMPLETO - 863 PARCHES
cat("=== INICIANDO ANÁLISIS COMPLETO ===\n")
cat("Parches a procesar:", nrow(habitat_nodes), "\n")
start_time <- Sys.time()
cat("Hora de inicio:", as.character(start_time), "\n")
```

```
cat("Estimación: 1.5-2.5 horas\n")
cat("====================================\n")
test <- MK_Focal_nodes(nodes = habitat_nodes,
                       id = "Id",
                       attribute = NULL,
                       raster_attribute = NULL,
                       fun_attribute = NULL,
                       distance = list(type = "edge", keep = 0.1),
                       metric = "PC",
                       probability = 0.5,
                       distance_thresholds = 10000,
                       search_buffer = 20000,
                       fragmentation = FALSE,
                       parallel = 4,
                       intern = FALSE)
end_time <- Sys.time()
tiempo_total <- round(as.numeric(difftime(end_time, start_time, units = "hours")), 2)
cat("=== ANÁLISIS COMPLETADO ===\n")
cat("Tiempo total:", tiempo_total, "horas\n")
# Guardar resultado inmediatamente
save(test, file = "resultado_PC_863_parches.RData")
cat("Resultado guardado\n")
# Visualización con todos los parches
breaks <- classInt::classIntervals(test$PC, n = 9, style = "jenks")
PC <- test %>%
  mutate(dPC_q = cut(PC,
                     breaks = breaks$brks,
                     include.lowest = TRUE,
                     dig.lab = 5))
# Mapa final
ggplot() +
  geom_sf(data = paisaje, fill = NA, color = "black") +
  geom_sf(data = PC, aes(fill = dPC_q), color = "black", size = 0.1) +
  scale_fill_brewer(palette = "RdYlGn", direction = 1, name = "PC (jenks)") +
```

```
  theme_minimal() +
  labs(
    title = "PC en paisajes focales",
    fill = "PC"
  ) +
  theme(
    legend.position = "right",
    plot.title = element_text(hjust = 0.5)
  )
# Calcular los intervalos de Jenks para strength
breaks <- classInt::classIntervals(test$dPC, n = 9, style = "jenks")
# Crear una nueva variable categórica con los intervalos
PC <- test %>%
  mutate(dPC_q = cut(dPC,
                     breaks = breaks$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 = PC, aes(fill = dPC_q), color = "black", size = 0.1) +
  scale_fill_brewer(palette = "RdYlGn", direction = 1, name = "dPC (jenks)") +
  theme_minimal() +
  labs(
    title = "dPC",
    fill = "dPC"
  ) +
  theme(
    legend.position = "right",
    plot.title = element_text(hjust = 0.5)
  )
# Calcular los intervalos de Jenks para strength
breaks <- classInt::classIntervals(test$dPCintra, n = 9, style = "jenks")
```

```
# Crear una nueva variable categórica con los intervalos
PC <- test %>%
  mutate(dPC_q = cut(dPCintra,
                     breaks = breaks$brks,
                     include.lowest = TRUE,
                     dig.lab = 5))
# Graficar en ggplot2 usando las clases Jenks
ggplot() +
  geom_sf(data = paisaje, fill = NA, color = "black") +
  geom_sf(data = PC, aes(fill = dPC_q), color = "black", size = 0.1) +
  scale_fill_brewer(palette = "RdYlGn", direction = 1, name = "dPCintra (jenks)") +
  theme_minimal() +
  labs(
    title = "dPCintra",
    fill = "dPCintra"
  ) +
  theme(
    legend.position = "right",
    plot.title = element_text(hjust = 0.5)
  )
# Calcular los intervalos de Jenks para strength
breaks <- classInt::classIntervals(test$dPCflux, n = 9, style = "jenks")
# Crear una nueva variable categórica con los intervalos
PC <- test %>%
  mutate(dPC_q = cut(dPCflux,
                     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 = PC, aes(fill = dPC_q), color = "black", size = 0.1) +
  scale_fill_brewer(palette = "RdYlGn", direction = 1, name = "dPCflux (jenks)") +
  theme_minimal() +
```

```
  labs(
    title = "dPCflux",
    fill = "dPCflux"
  ) +
  theme(
    legend.position = "right",
    plot.title = element_text(hjust = 0.5)
  )
# Calcular los intervalos de Jenks para strength
breaks <- classInt::classIntervals(test$dPCconnector, n = 9, style = "jenks")
# Crear una nueva variable categórica con los intervalos
PC <- test %>%
  mutate(dPC_q = cut(dPCconnector,
                     breaks = breaks$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 = PC, aes(fill = dPC_q), color = "black", size = 0.1) +
  scale_fill_brewer(palette = "RdYlGn", direction = 1, name = "dPCconnector (jenks)") +
  theme_minimal() +
  labs(
    title = "dPCconnector",
    fill = "dPCconnector"
  ) +
  theme(
    legend.position = "right",
    plot.title = element_text(hjust = 0.5)
  )
# Calcular los intervalos de Jenks para strength
breaks <- classInt::classIntervals(test$IComp, n = 9, style = "jenks")

# Crear una nueva variable categórica con los intervalos
```

```
PC <- test %>%
  mutate(dPC_q = cut(IComp,
                     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 = PC, aes(fill = dPC_q), color = "black", size = 0.1) +
  scale_fill_brewer(palette = "RdYlGn", direction = 1, name = "IComp (jenks)") +
  theme_minimal() +
  labs(
    title = "Índice de Conectividad Compuesto",
    fill = "IComp"
  ) +
  theme(
    legend.position = "right",
    plot.title = element_text(hjust = 0.5)
  )
################################•          Prioridad de enlaces (eliminación y cambio de enlaces)#######
library(ggplot2)
library(sf)
library(terra)
library(raster)
library(Makurhini)
library(RColorBrewer)
# Cargar tus datos
habitat_nodes <- read_sf("C:/Data_ecologia/Trabajo_final/Final2/Bosque_Caatinga.shp")
paisaje <- read_sf("C:/Data_ecologia/Trabajo_final/Final2/Paisaje_Estudio.shp")
# Verificar sistemas de coordenadas
print("CRS del paisaje:")
print(st_crs(paisaje))
print("CRS de habitat_nodes:")
print(st_crs(habitat_nodes))
```

```
# ESTABLECER CRS CORRECTO PARA BRASIL
target_crs <- "EPSG:31983"  # SIRGAS 2000 / UTM zone 23S para Brasil (en metros)
# Transformar ambos datasets al CRS correcto
paisaje <- st_transform(paisaje, target_crs)
habitat_nodes <- st_transform(habitat_nodes, target_crs)
# Verificar las coordenadas después de la transformación
print("Bbox del paisaje después de transformación:")
print(st_bbox(paisaje))
# Asegurar que ambos tengan el mismo CRS
if(st_crs(paisaje) != st_crs(habitat_nodes)) {
  habitat_nodes <- st_transform(habitat_nodes, st_crs(paisaje))
}
# Mapa inicial para verificar ubicación
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()) +
  labs(title = "Verificación de ubicación")
set.seed(2)
parches_ejemplo <- habitat_nodes[sample(1:nrow(habitat_nodes), 20),]
parches_ejemplo$Id <- 1:20
# CREAR RASTER DE RESISTENCIA CON RESOLUCIÓN 200m
bbox_area <- st_bbox(paisaje)
# Crear raster que cubra tu área de estudio con resolución de 200m
resistance_local <- terra::rast(
  xmin = bbox_area["xmin"],
  xmax = bbox_area["xmax"],
  ymin = bbox_area["ymin"],
  ymax = bbox_area["ymax"],
  resolution = c(200, 200),  # 200 metros de resolución
```

```
  crs = target_crs
)
# Llenar con valores de resistencia aleatorios
set.seed(42)
values(resistance_local) <- runif(ncell(resistance_local), 20, 100)
# Convertir a dataframe para ggplot
raster_df <- as.data.frame(resistance_local, xy = TRUE)
names(raster_df) <- c("x", "y", "value")
raster_df <- raster_df[!is.na(raster_df$value), ]
# MAPA FINAL CON RASTER DE RESISTENCIA 200m
ggplot() +
  geom_tile(data = raster_df, aes(x = x, y = y, fill = value), alpha = 0.8) +
  geom_sf(data = paisaje, fill = NA, color = "black", linewidth = 1) +
  geom_sf(data = habitat_nodes, fill = "forestgreen", color = "darkgreen",
          linewidth = 0.1, alpha = 0.9) +
  scale_fill_gradientn(
    colors = c("#000004FF", "#1B0C42FF", "#4B0C6BFF", "#781C6DFF",
               "#A52C60FF", "#CF4446FF", "#ED6925FF", "#FB9A06FF",
               "#F7D03CFF", "#FCFFA4FF"),
    name = "value"
  ) +
  theme_minimal() +
  labs(
    title = "Mapa de Resistencia - Bosque Caatinga, Brasil"
  ) +
  theme(
    axis.title.x = element_blank(),
    axis.title.y = element_blank(),
    plot.title = element_text(hjust = 0.5, size = 14, face = "bold"),
    legend.position = "right"
  )
library(gdistance)
# Convertir raster de terra a raster package para gdistance
resistance_matrix <- raster(resistance_local)
```

```
# A los valores NA les asignamos un alto valor para evitar que pasen por ahí
resistance_matrix[is.na(resistance_matrix)] <- 1000
# Estimamos la matriz de transición
tr <- transition(resistance_matrix, function(x) 1/mean(x), 8)
# Hacemos una corrección para los movimientos en diagonal
tr <- geoCorrection(tr, type = "c")
# Estimamos el centroide de nuestros parches
centroides <- st_centroid(parches_ejemplo, of_largest_polygon = TRUE)
centroides <- st_coordinates(centroides)
# Loop CORREGIDO para estimar corredores entre parches
rutas_list <- list()
for (i in 1:(nrow(centroides) - 1)) {
  cat(paste0("Procesando parche ", i, " de ", nrow(centroides), "\n"))
  rutas_temp <- list()
  for (j in (i + 1):nrow(centroides)) {
    tryCatch({
      ruta <- shortestPath(tr, centroides[i,], centroides[j,], output = "SpatialLines")
      if (!is.null(ruta)) {
        ruta <- st_as_sf(ruta)
        st_crs(ruta) <- st_crs(habitat_nodes)
        ruta$from <- i
        ruta$to <- j
        rutas_temp[[length(rutas_temp) + 1]] <- ruta
      }
    }, error = function(e) {
      cat("Error en ruta", i, "->", j, ":", e$message, "\n")
    })
  }
  if (length(rutas_temp) > 0) {
    rutas_list[[i]] <- do.call(rbind, rutas_temp)
  }
}
```

```
# Combinar todas las rutas
rutas_mc <- do.call(rbind, rutas_list[!sapply(rutas_list, is.null)])
# Mapa final con corredores
ggplot() +
  geom_sf(data = paisaje, fill = NA, color = "black") +
  geom_sf(data = rutas_mc, color = "black", linewidth = 0.5) +
  geom_sf(data = parches_ejemplo, fill = "forestgreen", color = "darkgreen",
          linewidth = 0.5) +
  theme_minimal() +
  labs(
    title = "Corredores potenciales"
  ) +
  theme(
    plot.title = element_text(hjust = 0.5),
    axis.title.x = element_blank(),
    axis.title.y = element_blank()
  )
#Distancia efectiva promedio como umbral de distancia
Effec_mean <- mean(resistance_matrix[], na.rm = TRUE) * 10000 # 10km
#Aplicamos la función
delta <- MK_dPCIIC_links(nodes = parches_ejemplo,
                         attribute = NULL,
                         area_unit = "ha",
                         distance = list(type = "least-cost",
                                         resistance = resistance_matrix),
                         removal = TRUE,
                         metric = "PC",
                         probability = 0.5,
                         distance_thresholds = round(Effec_mean),
                         parallel = NULL,
                         parallel_mode = 0,
                         intern = TRUE)
head(delta)
```

```
#Existen otras formas, pero crearé un nuevo ID
delta$ID_nuevo <- paste0(delta$Destination, "_", delta$Source)
#Guardo las rutas en un objeto nuevo para tener de respaldo mi vector original
rutas_mc2 <- rutas_mc
rutas_mc2$ID_nuevo <- paste0(rutas_mc2$from, "_", rutas_mc$to)
#Aplicar merge
rutas_mc2 <- merge(rutas_mc2, delta, by = "ID_nuevo")
rutas_mc2
library(ggplot2)
library(classInt)
library(dplyr)
# Calcular los intervalos de Jenks para strength
breaks <- classInt::classIntervals(rutas_mc2$dPC_removal, n = 5, style = "quantile")
# Crear una nueva variable categórica con los intervalos
rutas_mc2 <- rutas_mc2 %>%
  mutate(dPC_q = cut(dPC_removal,
                     breaks = breaks$brks,
                     include.lowest = TRUE,
                     dig.lab = 5))
# Graficar usando ggplot2 y colores de ColorBrewer
ggplot() +
  geom_sf(data = paisaje, fill = NA, color = "black") +
  geom_sf(data = rutas_mc2, aes(color = dPC_q), size = 0.5, linewidth = 1) +
  scale_color_brewer(palette = "RdYlBu", direction = -1, name = "dPC remove (Q)") +
  geom_sf(data = parches_ejemplo, aes(color = "Habitat nodes"),
          fill = "forestgreen", color = NA, linewidth = 0.5) +
  theme_minimal() +
  labs(
    title = "Priorización de enlaces (remove)",
    fill = "dPC"
  ) +
  theme(
    legend.position = "right",
    plot.title = element_text(hjust = 0.5)
```

```
 )
distancias <- distancefile(parches_ejemplo,
                           id = "Id",
                           type = "least-cost",
                           resistance = resistance_matrix,
                           pairwise = FALSE)
#No de enlaces
n <- 30
# Numero total de elementos
total_elements <- length(distancias)
# seleccion aleatoria
set.seed(4)
rand_idx <- sample(1:total_elements, n)
# obtener posiciones en la matriz de distancias
rand_positions <- arrayInd(rand_idx, .dim = dim(distancias))
# A esos enlaces les reduciremos un 40% de su valor
distancias_restauracion <- distancias
reduccion <- (40*distancias_restauracion[rand_positions])/100
distancias_restauracion[rand_positions] <- distancias_restauracion[rand_positions] - reduccion
distancias[rand_positions]
distancias_restauracion[rand_positions]
#Distancia efectiva promedio como umbral de distancia
Effec_mean <- mean(resistance_matrix[], na.rm = TRUE) * 10000 # 10km
#[1] 5229259
#Aplicamos la función
delta <- MK_dPCIIC_links(nodes = parches_ejemplo,
                         attribute = NULL,
                         area_unit = "ha",
                         distance = distancias,
                         removal = TRUE,
                         change = distancias_restauracion,
                         metric = "PC",
                         probability = 0.5,
                         distance_thresholds = round(Effec_mean),
```

```
                    parallel = NULL,
                    parallel_mode = 0,
                    intern = TRUE)
names(delta)
head(delta$Link_change_importances_d5229259)
change_corr <- delta$Link_change_importances_d5229259
change_corr$ID_nuevo <- paste0(change_corr$Destination, "_", change_corr$Source)
#Por precaución guardamos los corredores en otro objeto y generamos el ID
rutas_mc3 <- rutas_mc
rutas_mc3$ID_nuevo <- paste0(rutas_mc3$from, "_", rutas_mc3$to)
#Unimos
rutas_mc3 <- merge(rutas_mc3, change_corr, by = "ID_nuevo")
rutas_mc3
# Calcular los intervalos de Jenks para strength
breaks <- classInt::classIntervals(rutas_mc3$dPC_change, n = 5, style = "jenks")
# Crear una nueva variable categórica con los intervalos
rutas_mc3 <- rutas_mc3 %>%
  mutate(dPC_q = cut(dPC_change,
                     breaks = breaks$brks,
                     include.lowest = TRUE,
                     dig.lab = 5))
# Graficar usando ggplot2 y colores de ColorBrewer
ggplot() +
  geom_sf(data = paisaje, fill = NA, color = "black") +
  geom_sf(data = rutas_mc3, aes(color = dPC_q), size = 0.5, linewidth = 1) +
  scale_color_brewer(palette = "RdYlBu", direction = 1, name = "dPC change (jenks)") +
  geom_sf(data = parches_ejemplo, aes(color = "Habitat nodes"),
          fill = "forestgreen", color = NA, linewidth = 0.5) +
  theme_minimal() +
  labs(
    title = "Priorización de enlaces (change)",
    fill = "dPC"
  ) +
  theme(
```

```
    legend.position = "right",
    plot.title = element_text(hjust = 0.5)
  )
library(ggplot2)
library(sf)
library(terra)
library(raster)
library(Makurhini)
library(RColorBrewer)
library(gdistance)
library(dplyr)
library(classInt)
# ============================
# 1) Cargar datos
# ============================
habitat_nodes <- read_sf("C:/Data_ecologia/Trabajo_final/Final2/Bosque_Caatinga.shp")
paisaje <- read_sf("C:/Data_ecologia/Trabajo_final/Final2/Paisaje_Estudio.shp")
# Verificar CRS
message("CRS del paisaje:"); print(st_crs(paisaje))
message("CRS de habitat_nodes:"); print(st_crs(habitat_nodes))
# CRS para Brasil (metros)
target_crs <- "EPSG:31983"
# Transformar ambos
paisaje <- st_transform(paisaje, target_crs)
habitat_nodes <- st_transform(habitat_nodes, target_crs)
# ============================
# 2) Verificación visual
# ============================
ggplot() +
  geom_sf(data = paisaje, fill = NA, color = "black") +
  geom_sf(data = habitat_nodes, fill = "forestgreen", color = "darkgreen", linewidth = 0.5) +
  theme_minimal() +
  labs(title = "Verificación de ubicación") +
  theme(axis.title.x = element_blank(), axis.title.y = element_blank())
```

```
# ==========================
# 3) Parches ejemplo (ID como CHARACTER y como PRIMERA columna)
# ==========================
set.seed(2)
parches_ejemplo <- habitat_nodes[sample(1:nrow(habitat_nodes), 20), ]
# --- Asegura que exista 'Id' y que sea character ---
if (!"Id" %in% names(parches_ejemplo)) {
  parches_ejemplo$Id <- as.character(1:nrow(parches_ejemplo))
} else {
  parches_ejemplo$Id <- as.character(parches_ejemplo$Id)
}
# --- REORDENAR columnas sin dplyr::select (base R, funciona con sf) ---
cols <- names(parches_ejemplo)
neworder <- c("Id", cols[cols != "Id"])
parches_ejemplo <- parches_ejemplo[, neworder]
# --- Si por alguna razón anterior no funciona, descomenta el fallback:
# geom_temp <- st_geometry(parches_ejemplo)
# df_temp <- st_drop_geometry(parches_ejemplo)
# df_temp <- df_temp[, c("Id", setdiff(names(df_temp), "Id"))]
# parches_ejemplo <- st_as_sf(df_temp, geometry = geom_temp, crs = st_crs(patches_ejemplo))
# ==========================
# ==========================
# 4) Crear raster resistencia 200m
# ==========================
bbox_area <- st_bbox(paisaje)
resistance_local <- terra::rast(
  xmin = bbox_area["xmin"], xmax = bbox_area["xmax"],
  ymin = bbox_area["ymin"], ymax = bbox_area["ymax"],
  resolution = c(200, 200), crs = target_crs
)
set.seed(42)
values(resistance_local) <- runif(ncell(resistance_local), 20, 100)
# Mapa resistencia (opcional)
raster_df <- as.data.frame(resistance_local, xy = TRUE)
```

```
names(raster_df) <- c("x","y","value")
raster_df <- raster_df[!is.na(raster_df$value), ]
ggplot() +
  geom_tile(data = raster_df, aes(x = x, y = y, fill = value), alpha = 0.8) +
  geom_sf(data = paisaje, fill = NA, color = "black") +
  geom_sf(data = habitat_nodes, fill = "forestgreen", color = "darkgreen", linewidth = 0.2) +
  scale_fill_gradientn(colors = c("#000004FF","#1B0C42FF","#4B0C6BFF","#781C6DFF",
                                  "#A52C60FF","#CF4446FF","#ED6925FF","#FB9A06FF",
                                  "#F7D03CFF","#FCFFA4FF"), name = "value") +
  theme_minimal() + labs(title = "Mapa de Resistencia - Bosque Caatinga")
# ============================
# 5) gdistance – rutas (usando Id reales para 'from' y 'to')
# ============================
resistance_matrix <- raster(resistance_local)   # convert terra->raster
resistance_matrix[is.na(resistance_matrix)] <- 1000
tr <- transition(resistance_matrix, function(x) 1/mean(x), 8)
tr <- geoCorrection(tr, type = "c")
# centroides en el orden de parches_ejemplo
centroides_sf <- st_centroid(parches_ejemplo, of_largest_polygon = TRUE)
centroides_coords <- st_coordinates(centroides_sf)
rutas_list <- list()
for (i in 1:(nrow(centroides_coords)-1)) {
  rutas_temp <- list()
  for (j in (i+1):nrow(centroides_coords)) {
    ruta_sp <- tryCatch(
      shortestPath(tr, centroides_coords[i,], centroides_coords[j,], output = "SpatialLines"),
      error = function(e) NULL
    )
    if (!is.null(ruta_sp)) {
      ruta_sf <- st_as_sf(ruta_sp)
      st_crs(ruta_sf) <- st_crs(parches_ejemplo)
      # store origin/destination using los Id reales (no índices)
      ruta_sf$from <- parches_ejemplo$Id[i]
      ruta_sf$to   <- parches_ejemplo$Id[j]
```

```
      rutas_temp[[length(rutas_temp)+1]] <- ruta_sf
    }
  }
  if (length(rutas_temp)>0) rutas_list[[i]] <- do.call(rbind, rutas_temp)
}
rutas_mc <- do.call(rbind, rutas_list[!sapply(rutas_list,is.null)])
# ============================
# 6) dPC removal (sin 'id' — MK_dPCIIC_links no acepta 'id' como argumento)
# ============================
Effec_mean <- mean(resistance_matrix[], na.rm=TRUE) * 10000
delta_rem <- MK_dPCIIC_links(
  nodes = parches_ejemplo,          # sf con Id como primera columna
  attribute = NULL,
  area_unit = "ha",
  distance = list(type = "least-cost", resistance = resistance_matrix),
  removal = TRUE,
  metric = "PC",
  probability = 0.5,
  distance_thresholds = round(Effec_mean),
  parallel = NULL,
  parallel_mode = 0,
  intern = TRUE
)
delta_rem$Destination <- as.character(delta_rem$Destination)
delta_rem$Source      <- as.character(delta_rem$Source)
rem_long <- rbind(
  transform(delta_rem[, c("Destination","Source","dPC_removal")],
            ID_nuevo = paste0(Destination, "_", Source)),
  transform(delta_rem[, c("Destination","Source","dPC_removal")],
            ID_nuevo = paste0(Source, "_", Destination))
)
rutas_mc2 <- rutas_mc
rutas_mc2$ID_nuevo <- paste0(as.character(rutas_mc2$from), "_", as.character(rutas_mc2$to))
```

```
rutas_mc2 <- rutas_mc2 %>%
  left_join(rem_long[, c("ID_nuevo","dPC_removal")], by = "ID_nuevo") %>%
  filter(!is.na(dPC_removal))
rutas_mc2$dPC_removal <- as.numeric(rutas_mc2$dPC_removal)
vals <- rutas_mc2$dPC_removal
ncl <- min(5, max(2, length(unique(vals))))
brks <- try(classIntervals(vals, n = ncl, style = "quantile"), silent = TRUE)
if (inherits(brks, "try-error")) {
  brks <- list(brks = quantile(vals, probs = seq(0,1,length.out = ncl+1), na.rm = TRUE))
}
rutas_mc2$dPC_q <- cut(rutas_mc2$dPC_removal, breaks = brks$brks, include.lowest = TRUE)
ggplot() +
  geom_sf(data = paisaje, fill = NA, color = "black") +
  geom_sf(data = rutas_mc2, aes(color = dPC_q), linewidth = 1) +
  scale_color_brewer(palette = "RdYlBu", direction = -1, name = "dPC remove") +
  geom_sf(data = parches_ejemplo, fill = "forestgreen", color = "darkgreen", linewidth = 0.5) +
  theme_minimal() + labs(title = "Priorización de enlaces (remove)")
# ============================
# 7) Distancias restauración y dPC change (matriz con dimnames = Id)
# ============================
distancias <- distancefile(parches_ejemplo,
                           id = "Id",
                           type = "least-cost",
                           resistance = resistance_matrix,
                           pairwise = FALSE)
ids_chr <- parches_ejemplo$Id
dimnames(distancias) <- list(ids_chr, ids_chr)
distancias_restauracion <- distancias
set.seed(4)
cand <- which(row(distancias) != col(distancias))
sel <- sample(cand, min(30, length(cand)))
pos <- arrayInd(sel, .dim = dim(distancias))
reduccion <- 0.4 * distancias_restauracion[pos]
distancias_restauracion[pos] <- distancias_restauracion[pos] - reduccion
```

```
delta_ch <- MK_dPCIIC_links(
  nodes = parches_ejemplo,
  attribute = NULL,
  area_unit = "ha",
  distance = distancias,
  change = distancias_restauracion,
  removal = TRUE,
  metric = "PC",
  probability = 0.5,
  distance_thresholds = round(Effec_mean),
  parallel = NULL,
  parallel_mode = 0,
  intern = TRUE
)
tbl_name <- grep("^Link_change_importances_", names(delta_ch), value = TRUE)[1]
change_corr <- delta_ch[[tbl_name]]
change_corr$Destination <- as.character(change_corr$Destination)
change_corr$Source      <- as.character(change_corr$Source)
change_corr$dPC_change  <- as.numeric(change_corr$dPC_change)
change_long <- rbind(
  transform(change_corr[, c("Destination","Source","dPC_change")],
            ID_nuevo = paste0(Destination, "_", Source)),
  transform(change_corr[, c("Destination","Source","dPC_change")],
            ID_nuevo = paste0(Source, "_", Destination))
)
rutas_mc3 <- rutas_mc
rutas_mc3$ID_nuevo <- paste0(as.character(rutas_mc3$from), "_", as.character(rutas_mc3$to))
rutas_mc3 <- rutas_mc3 %>%
  left_join(change_long[, c("ID_nuevo","dPC_change")], by = "ID_nuevo") %>%
  filter(!is.na(dPC_change))
vals2 <- rutas_mc3$dPC_change
ncl2 <- min(5, max(2, length(unique(vals2))))
brks2 <- try(classIntervals(vals2, n = ncl2, style = "jenks"), silent = TRUE)
if (inherits(brks2, "try-error")) {
```

```
  brks2 <- list(brks = quantile(vals2, probs = seq(0,1,length.out = ncl2+1), na.rm = TRUE))
}
rutas_mc3$dPC_q <- cut(rutas_mc3$dPC_change, breaks = brks2, include.lowest = TRUE)
ggplot() +
  geom_sf(data = paisaje, fill = NA, color = "black") +
  geom_sf(data = rutas_mc3, aes(color = dPC_q), size = 0.5, linewidth = 1) +
  scale_color_brewer(palette = "RdYlBu", direction = 1, name = "dPC change (jenks)") +
  geom_sf(data = parches_ejemplo, aes(color = "Habitat nodes"),
          fill = "forestgreen", color = NA, linewidth = 0.5) +
  theme_minimal() +
  labs(
    title = "Priorización de enlaces (change)",
    fill = "dPC"
  ) +
  theme(
    legend.position = "right",
    plot.title = element_text(hjust = 0.5)
  )
##########################################ECA (Equivalent Connected Area) y dECA###############
library(ggplot2)
library(sf)
library(terra)
library(sp)
library(raster)
library(igraph)
library(Makurhini)
library(RColorBrewer)
# Cargar tus datos
habitat_nodes <- read_sf("C:/Data_ecologia/Trabajo_final/Final2/Bosque_Caatinga.shp")
paisaje <- read_sf("C:/Data_ecologia/Trabajo_final/Final2/Paisaje_Estudio.shp")
# Verificar sistemas de coordenadas
print("CRS del paisaje:")
print(st_crs(paisaje))
print("CRS de habitat_nodes:")
```

```
print(st_crs(habitat_nodes))
# ESTABLECER CRS CORRECTO PARA BRASIL
target_crs <- "EPSG:31983"  # SIRGAS 2000 / UTM zone 23S para Brasil (en metros)
# Transformar ambos datasets al CRS correcto
paisaje <- st_transform(paisaje, target_crs)
habitat_nodes <- st_transform(habitat_nodes, target_crs)
# Verificar las coordenadas después de la transformación
print("Bbox del paisaje después de transformación:")
print(st_bbox(paisaje))
# Asegurar que ambos tengan el mismo CRS
if(st_crs(paisaje) != st_crs(habitat_nodes)) {
  habitat_nodes <- st_transform(habitat_nodes, st_crs(paisaje))
}
# Mapa inicial para verificar ubicación
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()) +
  labs(title = "Verificación de ubicación")
ECA1 <- MK_dPCIIC(nodes = habitat_nodes,
                  attribute = NULL,
                  area_unit = "ha",
                  distance = list(type = "edge", keep = 0.1),
                  LA = NULL,
                  metric = "PC",
                  probability = 0.5,
                  onlyoverall = TRUE,
                  distance_thresholds = 10000,
                  intern = FALSE) #10 km
ECA1
ECA1[2,2]
```

```
ECA2 <- ECA1[2,2]

paisaje_area <- st_area(paisaje)

paisaje_area <- unit_convert(paisaje_area, "m2", "ha")

#ECA normalizado respecto al área del paisaje

(ECA2*100)/paisaje_area

#> [1] 41.31591

bosque_area <- st_area(habitat_nodes)

bosque_area <- unit_convert(bosque_area,  "m2", "ha")

bosque_area <- sum(bosque_area)

#ECA normalizado respecto al área del bosque

(ECA2*100)/bosque_area

#> [1] 89.58149

paisaje <- read_sf("C:/Data_ecologia/Trabajo_final/Final2/Paisaje_Estudio.shp")

Max_atributo <- as.numeric(st_area(area_estudio)) * 0.0001 # Hectáreas

Max_atributo

dECA_test <- MK_dECA(nodes= lista_parches,

                     attribute = NULL,

                     area_unit = "ha",

                     distance = list(type= "edge", keep = 0.1),

                     metric = "PC",

                     probability = 0.05,

                     distance_thresholds = 5000,

                     LA = Max_atributo,

                     plot= c("2018"),

                     intern = FALSE)
```