Muestreo Adaptativo

Técnicas de Muestreo I: Proyecto Final

Christian Badillo Luis Nuñez Luz Maria Santana Sealtiel Pichardo

Tabla de contenidos

1 Introducción

Durante el curso de Técnicas de muestreo I 2025-1, se abordaron diferentes tipos de muestreo como muestreo aleatorio simple, estratificado, sistemático, de conglomerados, bietápico o con probabilidad proporcional al tamaño. Sin embargo, en miembros con individuos raros o escasos, los métodos anteriores pueden tener dificultades.

En general, los métodos estadísticos adaptativos se caracterizan por ajustar y redirigir la investigación hacia el procedimiento más adecuado, tomando de referencia a los datos que se están recolectando (O' Gorman, 2004). Los muestreos adaptativos surgen como respuesta a situaciones de incertidumbre respecto a la forma de la distribución de una medida de interés y a las características de ciertas miembros, las cuales podrían ser de difícil acceso o para las cuales simplemente es imposible determinar su locación específica como para formar estratos (Thompson, 2012), para los cuales las técnicas de muestreo tradicional resultan en costos elevados y estimaciones imprecisas. Este es el caso de miembros con eventos raros o grupos que se encuentran agrupados en sectores, tales como cardúmenes, ciertas especies de plantas en regiones específicas, en la búsqueda de minerales o medición de características raras en miembros humanos (tales como la incidencia de enfermedades raras).

Los primeros trabajos que utilizaron este muestreo datan de aproximadamente 50 años con los trabajos de Abraham Wlad en 1947, pero que después tuvieron un mayor impacto en el campo de la biología, con aplicaciones en encuestas de plantas y vida salvaje realizadas por S.K. Thompson. Además, este muestreo se ha utilizado en encuestas de especies como árboles raros, mejillones de agua dulce y ranas de cola (Thompson & Ramsey, 1983; Roesch, 1993; Box et al., 2002; Vesely & Stam, 2001).

Existe diversas variantes del muestreo adaptativo, puede ser por conglomerados, doble adaptativo por conglomerados, también se puede aplicar al muestreo estratificado, además de que otras técnicas como el muestreo inverso y secuencial pueden ser considerados como métodos de muestreo adaptativo; todos ellos tienen la ventaja de que maximizan la precisión de las estimaciones al usar un menor volumen de datos debido al ajuste dinámico que se hace de la muestra en función de la información obtenida durante la recolección de los datos (Seber & Salehi, 2012; Pereyra & Vaira, 2021). A continuación, se describirá el muestreo adaptativo en conglomerados.

2 Muestreo adaptativo en conglomerados

Para conocer un poco sobre cómo funciona este tipo de muestreo, pensemos en la siguiente situación:

Nos interesa conocer la cantidad que hay de un determinado pez en una zona oceánica. Comenzamos por dividir el territorio como una cuadrícula de *N* cuadrados e indicamos un número de cuadros iniciales n_1 de territorio que se explorarán, escogemos 4 cuadrantes y se determina un número C, por ejemplo 5. El muestreo consistiría en el siguiente procedimiento para cada cuadrante: Se cuenta cuántos peces de interés hay en el cuadrante, si este número de peces es mayor que el número C que se especificó (5) entonces se muestrean las zonas vecinas m_i de ese cuadrante (e.g. los cuadrantes norte, sur, este y oeste) y se vuelve a contar en cada una cuántos peces de interés hay y, si de nuevo se supera el valor C, se muestrean las zonas vecinas del cuadrante que cumplió con el

criterio y así sucesivamente hasta encontrar zonas que no cumplan con el criterio especificado. Si un cuadrante inicial no cumpliera con el criterio, entonces ya no se muestrean las zonas adyacentes de ese cuadrante.

Lo anterior representaría el esquema general del muestreo adaptativo en conglomerados, las zonas que cumplen con el criterio C se denominan unidades centrales y las que no se denominan unidades de borde (Seber & Salehi, 2012). Esto permite adaptar el muestreo de acuerdo con la variabilidad o densidad de datos que hay en una determinada zona, dando más interés a las que tienen mayor concentración/varianza y menos a las que no; esto resulta en un diseño en el cual el tamaño de la muestra se vuelve una variable aleatoria, ya que no se puede saber con antelación cuántos de los cuadrantes cumplirán con el criterio que se especifique y por ende cuántos datos se recolectarán (Seber & Salehi, 2012); además de tener estimadores que, de base, son sesgados (Thompson, 2012).

De acuerdo con Thompson (2012): Sea ψ_i la red que incluya a un elemento i y que sea m_i el número de unidades que compone a esa red (unidades de borde y centrales) entonces ω_i representa al promedio de las observaciones en cada red, esto es:

$$\omega_i = \frac{1}{m_i} \sum_{j \in \psi_i} y_i$$

Entonces el estimador insesgado para la media, cuando el muestreo es sin reemplazo, sería:

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} \omega_i$$

Con varianza:

$$var(\hat{\mu}) = \frac{N-n}{Nn(N-1)} \sum_{i=1}^{n} (w_i - \hat{\mu})^2$$

Y estimador de la varianza:

$$\hat{var}(\hat{\mu}) = \frac{N-n}{Nn(n-1)} \sum_{i=1}^{n} (\omega_i - \hat{\mu})^2$$

El muestreo adaptativo generará menor varianza que el estimador usual de muestreo aleatorio simple (m.a.s.) si y solo si la varianza dentro de las redes generadas es mayor que la varianza global de la población (Thompson, 2012). Es decir, si los valores altos, importantes o extremos se encuentran agrupados en redes específicas y no dispersos en todo el territorio, entonces será mejor el muestreo adaptativo ya que se aprovechará el realizar los pasos adicionales que involucran muestrear zonas vecinas.

2.1 Ejemplo 1.

Supongamos que se quiere realizar una muestra de un bosque para identificar el total de árboles de una especie rara, para ello se divide el bosque en parcelas (conglomerados) de igual tamaño. Para realizar el muestro adaptativo por conglomerados se simularon 100 parcelas agrupadas en 10 filas y columnas que componen al bosque. Se realizó un muestreo aleatorio simple inicial de $n = 5$ y después se procedio con un diseño de muestro adaptativo, el resiltado se puede observar en figura 1.

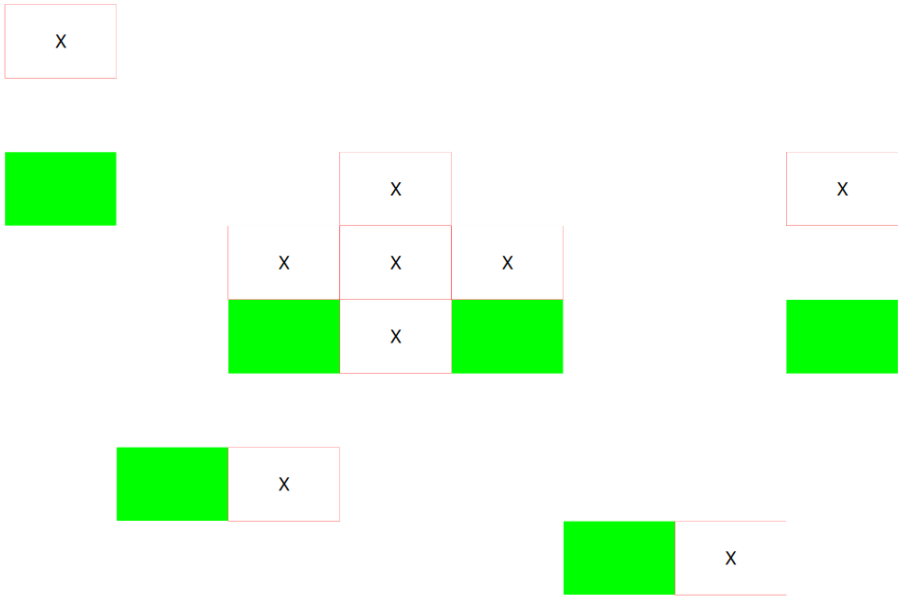

Figura 1: Muestreo Adaptativo.

En verde se observan (veáse figura 1) las parcelas con árboles “raros”, es decir, dentro de esas parcelas se encuentra algún árbol con las características buscadas o que sea difícil de encontrar. Al colocar una probabilidad del 5% (0.05) se asegura que sean pocas las parcelas que contengan un árbol raro.

Por otro lado las parcelas en color rojo contienen una X, lo que indica que han sido muestreadas, inicialmente se seleccionan 5 parcelas de manera aleatoria, sin considerar si tienen un árbol “raro”. Ahora, en el caso de que una de esas parcelas cuente con un árbol raro, el muestreo se expande, de tal forma que las parcelas que se encuentren arriba, abajo, a la izquierda o a la derecha, también son muestreadas y se agregan las unidades.

Al expandir el muestreo,se pueden encontrar agrupamientos o clusters al rededor de parcelas con los árboles de interés, además, al incluir las parcelas vecinas, aumenta la cantidad de muestras y se

mejora la representación de las unidades que contengan árboles “raros”.

2.2 Ejemplo 2.

Se generan datos simulados que siguen una distribución Poisson con $\lambda = 3$ en una cuadrícula de 20x20 ($N = 400$)

Visualizamos las zonas con mayor densidad de la característica de estudio, los cuadrados más azules representan a las zonas con mayor densidad y los blancos a los de menor:

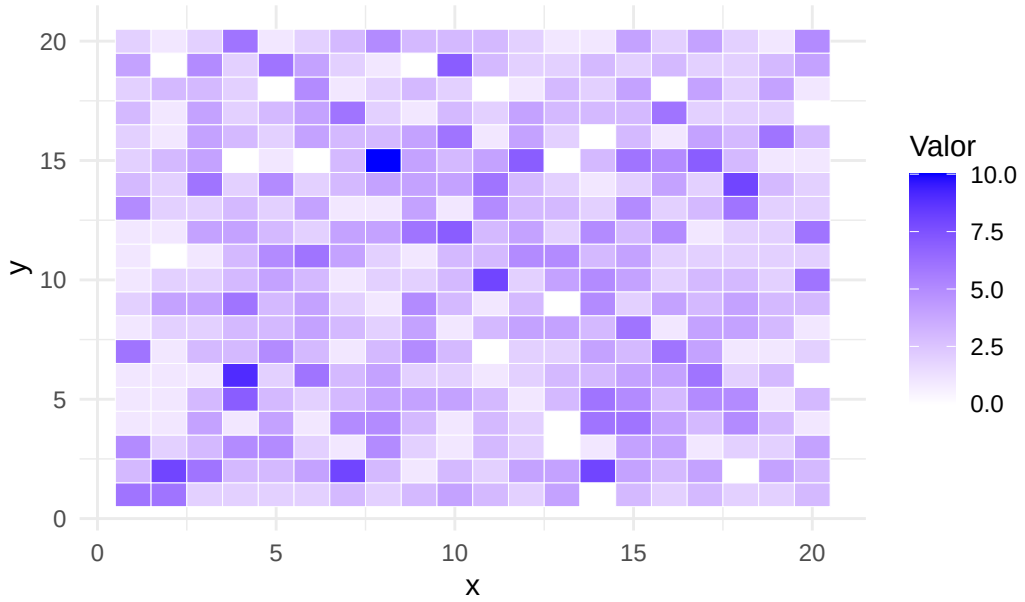

Figura 2: Población en clusters

Se definen los parámetros muestrales: n_1 (los cuadrantes iniciales) será igual a 5, el umbral adaptativo C será igual a 2 y se define la función que permite obtener a la primera muestra

```
n1 <- 5           # Tamaño inicial de la muestra
C <- 2     # Umbral adaptativo: expandir si value > C

neighbors <- function(x, y) {
  # Función para encontrar vecinos contiguos
  expand.grid(x = c(x - 1, x, x + 1), y = c(y - 1, y, y + 1)) %>%
    filter(!(x == x[1] & y == y[1])) %>% # Excluir la unidad central
    filter(x > 0 & y > 0 & x <= grid_size & y <= grid_size)
}

initial_sample <- population %>%
  sample_n(n1) %>%
  mutate(sampled = TRUE)
```

Se crea la función que permite simular el proceso a través del cual se evaluan las unidades respecto al criterio C y determinar si se deben o no muestrear los vecinos.

```
# Implementar expansión adaptativa
adaptive_sampling <- function(initial_sample, population, C) {
  sampled_units <- initial_sample

  while (TRUE) {
    # Buscar vecinos de las unidades ya muestreadas que no han sido visitadas
    new_units <- sampled_units %>%
      filter(sampled) %>%
      rowwise() %>%
      do(neighbors(.$x, .$y)) %>%
      ungroup() %>%
      distinct() %>%
      left_join(population, by = c("x", "y")) %>%
      filter(!x %in% sampled_units$x | !y %in% sampled_units$y) # Evitar duplicados

    # Agregar a la muestra aquellos que cumplen el criterio C
    new_sampled <- new_units %>%
      filter(value > C) %>%
      mutate(sampled = TRUE)

    # Si no hay nuevas unidades que expandir, terminamos
    if (nrow(new_sampled) == 0) break

    sampled_units <- bind_rows(sampled_units, new_sampled)
  }

  return(sampled_units)
}

final_sample <- adaptive_sampling(initial_sample, population, C)
```

Se visualiza cuáles cuadrantes se muestrearon de manera inicial (rojo) y cuáles como un producto del proceso adaptativo y que son centrales (amarillo).

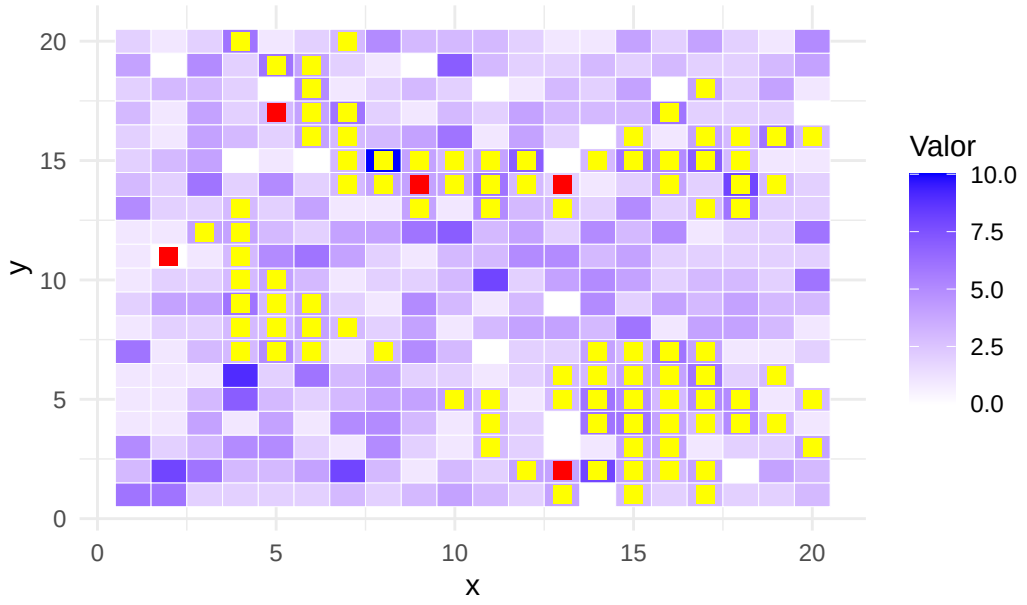

Figura 3: Muestreo Adaptativo de Clusters

Se realizan las estimaciones de la media y la varianza:

```r
# Identificar redes únicas
final_sample <- final_sample %>%
  group_by(network = cumsum(sampled))


# Identificar redes únicas
final_sample <- final_sample %>%
  group_by(network) %>%
  mutate(network_size = n()) %>%  # Tamaño de cada red (m_i)
  ungroup()

# Calcular w_i (promedio dentro de cada red)
omega <- final_sample %>%
  group_by(network) %>%
  summarize(omega_i = mean(value), .groups = "drop")

# Paso 3: Calcular estimador de la media (mu_hat)
mu_hat <- mean(omega$omega_i)

# Calcular estimador de la varianza de la media
# Primero, necesitamos el tamaño de la población y la muestra
N <- nrow(population)           # Total de unidades en la población
n <- length(unique(final_sample$network))  # Total de redes muestreadas

# Calcular la varianza estimada (var_hat_mu)
var_hat_mu <- ((N - n) / (N * n * (n - 1))) * sum((omega$omega_i - mu_hat)^2)

# Mostrar resultados
cat("Estimador de la media (mu_hat):", mu_hat, "\n")
```

```
Estimador de la media (mu_hat): 4.1
```

```
cat("Estimador de la varianza (var_hat_mu):", var_hat_mu, "\n")
```

```
Estimador de la varianza (var_hat_mu): 0.01583333
```

2.3 Ejemplo 3.

Para el ejemplo se construyeron datos que estuvieran en grupos aislados, posteriormente se realiza un tratamiento que nos permitió extraer una muestra adaptativa por clouster.

1. En la primera parte de este ejemplo se contruyeron las funciones necesarias para este método (véase apéndice):
    - La función `vecindad` que nos da los vecinos de un elemento seleccionado en una matriz, se considera periosidad en la frontera para facilidar los cálculos.
    - La función `crece` guarda las unidades que puede crecer una unidad incial, hasta tener unidades de borde.
    - Finalmente la función `redes` nos da las m_i y y_i de i unidades iniciales.
2. En la segunda parte del ejemplo, visualizamos los datos y contamos cuántos datos hay en cada unidad, este conteo lo realizamos en una matriz, los que nos permite llamar a cada elemento de la matriz como u_i
3. Finalmente aplicamos nuestras funciones y visualizamos como las muestras crecieron bajo la condición de la unidad, esta es si $u_i > C$ con C una constante. Agrupamos nuestros resultados en una tabla y calculamos el estimador de la media según lo visto en la explicación.

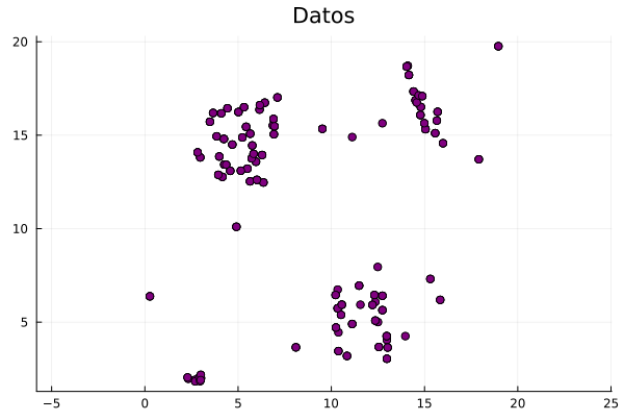

Figura 4: Datos para la simulación (Ejemplo 3).

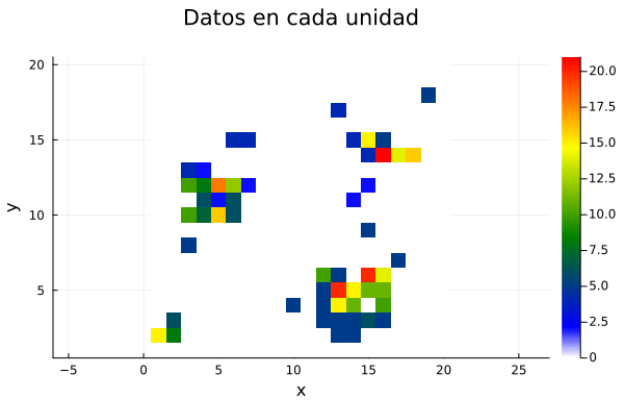

Figura 5: Conglomerados.

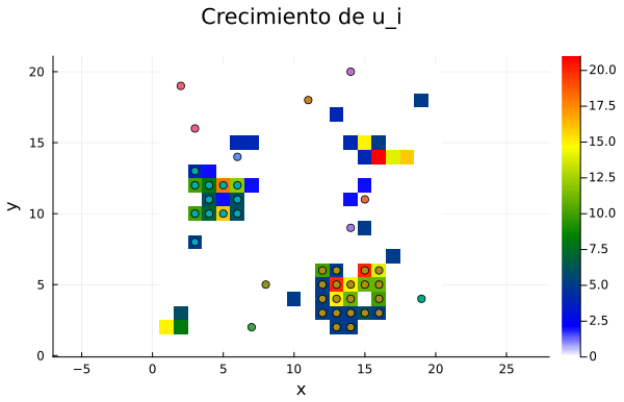

Figura 6: Muestreo (Ejemplo 3).

En la figura 5 se ve la formación de conglomerados cuando se usaron los datos. Visualizamos en el mapa de calor (véase figura 6) como avanzó cada unidad inicial, notaremos que solo una de ellas tenía vecinos que cumplían la condición de $u_i > C = 3$ y se detuvo cuando tenia unidades de borde.

Tabla 1: Datos de la Simulación (Ejemplo 3).

m_i	y_i	ω_i
1	[0]	0.000000
1	[0]	0.000000
1	[0]	0.000000
20	[10, 5, 11, 6, 14, 11, 5, 20, 15, 5, 11, 5, 20, 5, 15, 5, 5, 5, 5, 10]	9.400000
11	[10, 7, 6, 16, 8, 6, 10, 18, 6, 4, 12]	9.363636
1	[0]	0.000000
1	[0]	0.000000
1	[0]	0.000000
1	[0]	0.000000

Tabla 1: Datos de la Simulación (Ejemplo 3).

m_i	y_i	ω_i
1	[5]	5.000000
1	[0]	0.000000
1	[0]	0.000000
1	[0]	0.000000

Los resultados de la muestra simulada se encuentran en la tabla 1. La media estimada= 0.56580 con varianza=0.09267. Por lo tanto, el estimador total de la población sería $N \cdot \hat{\mu} = 0.597 \cdot 400 = 238.8 \approx 239$ de datos.

2.4 Ejemplo 4.

Suponga que existe una enfermedad rara respiratoria contagiosa en una cierta región de un país. Se desea estudiar el total de casos en un periodo determinado, para ello se toman como conglomerados las familias de la región. Dado que la enfermedad es contagiosa se espera que las familias relacionadas entre sí aumente la probabilidad de contagio y por ende la de observar casos. Por ello se realiza un muestro adaptativo teniendo en cuenta las relaciones entre las familias y como criterio de inclusión el que haya 1 caso.

En la figura 7 se muestran los resultados de la simulación, cada nodo representa una familia y tienen un identificador único, las aristas de los nodos indican las relaciones entre las familias (se controlo que el máximo fuese 4), la intensidad del color de relleno indica el número de casos en la familia. Los nodos con un borde rojo fueron tomados en muestra ya sea por medio del muestreo aleatorio simple inicial ($n = 10$) o por el proceso adaptativo. Las aristas en verde indican el esparcimiento de la muestra entre los nodos (es decir, si los nodos en muestra están relacionados entre sí) y sirven para visualizar las subredes formadas en el grafo original.

En la tabla 2 se muestran los datos obtenidos de la muestra seleccionada, en la tabla se muestra el identificador de la subred, los "nodos válidos" definidos como aquellos que cumplen el criterio (al menos 1 caso en la familia) y si ningún nodo de la red cumplían con el criterio se asigna 1 por defecto para evitar divisiones entre 0; y el total de casos en la red generada por los nodos así como el promedio de observaciones en cada red (ω_i).

Tabla 2: Datos Obtenidos de la Simulación (Ejemplo 4).

Red	No. Nodos Válidos	Casos	ω_i
0	19	22	1.157895
1	1	1	1.000000
2	2	3	1.500000
3	1	1	1.000000

La media de casos es $\mu = 0.1863$ con una varianza $s^2 = 0.0012103$ ($s = 0.0348$). Por tanto, el total

de casos estimado es 18.63 ($N = 100$) que podemos decir que hay 19 casos si se redondea hacia arriba, su varianza es 12.10271 ($s = 3.4789$). El estimador del total es menor a lo que en realidad hay (36 casos en total), pero esto puede ser debido a un inicio malo con el muestreo aleatorio simple, dado que varios nodos iniciales del MAS no tenían casos (38, 26) y por ende no se expandió la muestra en ellos o que no por azar la expansión en la muestra no alcanzo a las familias con más casos.

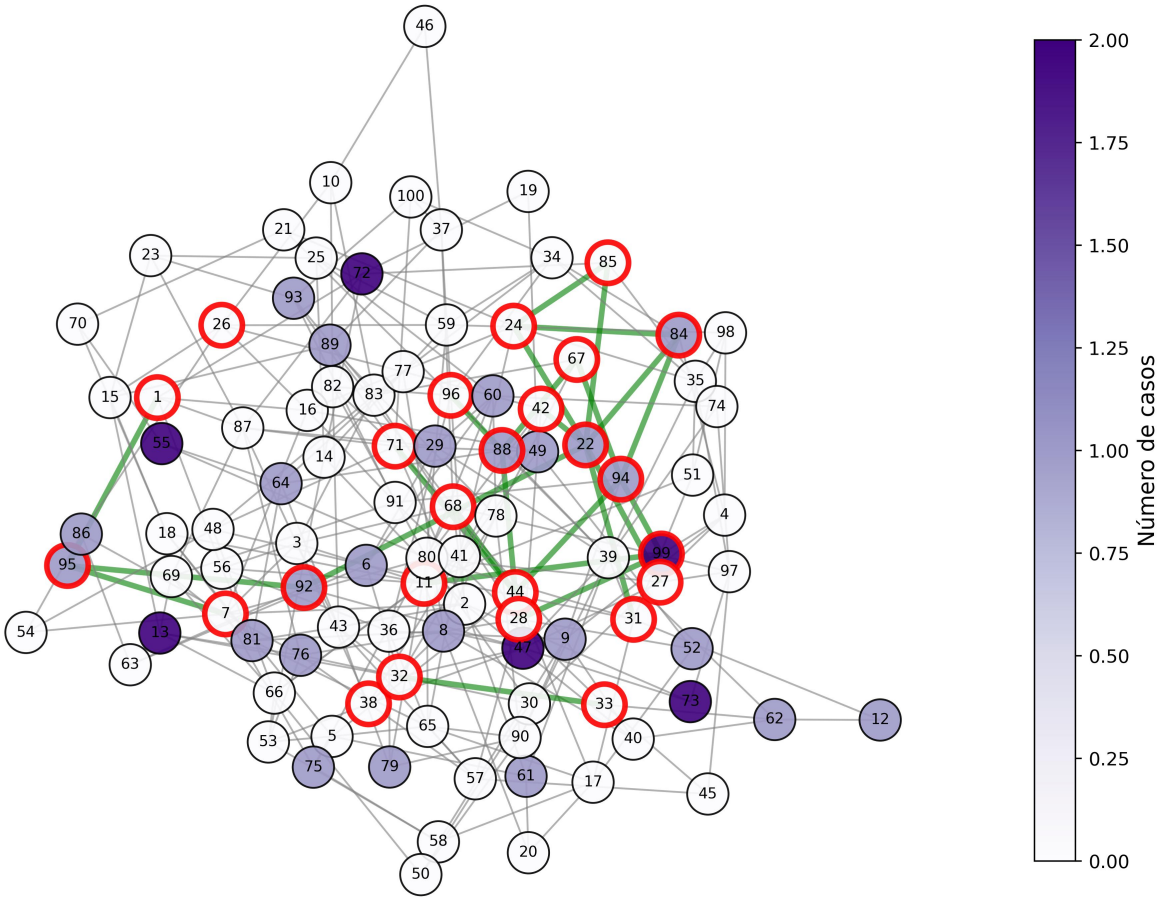

Figura 7: Muestreo Adaptativo en Familias.

2.5 Muestreo adaptativo por conglomerados: sistemático y por franjas

La principal diferencia entre el muestreo adaptativo por conglomerados y sus variantes de tipo sistemático y por franjas es la diferenciación entre unidades primarias y secundarias de muestreo. Las unidades primarias serán aquellas particiones iniciales de la cuadrícula de acuerdo a un patrón (por franjas o alguna regla sistemática como parejas de cuadrantes), las cuales serán todas muestreadas; y las unidades secundarias serán todos aquellos cuadrantes que se agregan al muestreo debido al

proceso adaptativo, estas serán contiguas a las unidades primarias pero pueden no tener un patrón sistemático.

Por ejemplo, de una cuadrícula de 10x10 podríamos seleccionar 3 líneas verticales (u horizontales) tal como se muestra en la siguiente figura, estas representarían a las unidades primarias y serían las franjas que estarán en la muestra inicial. Los puntos verdes indican zonas donde se cumpliría el criterio C que permite poner en marcha el proceso adaptativo.

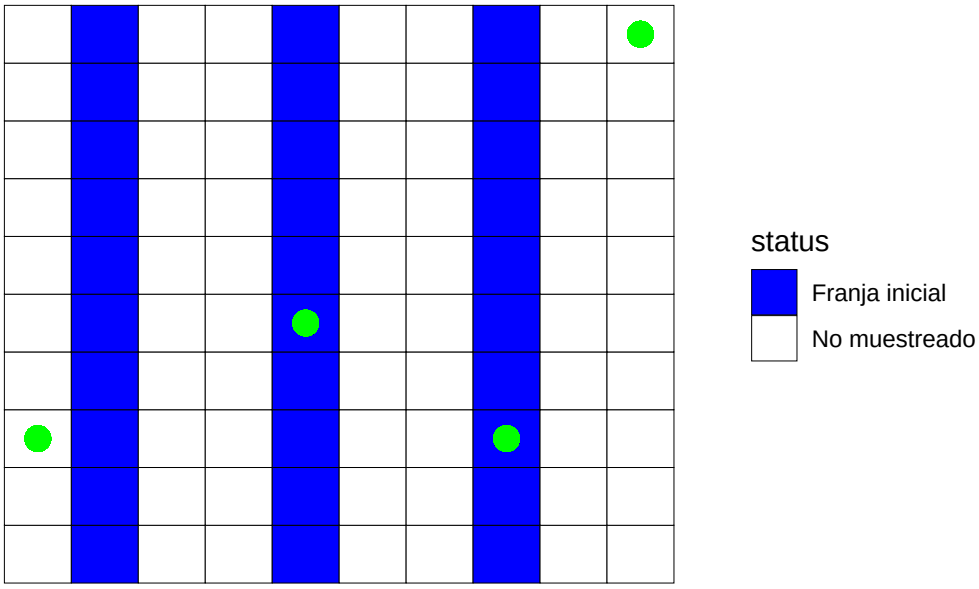

Figura 8: Muestreo inicial en franjas

Después de haber hecho la medición en las unidades primarias, entra el proceso adaptativo. Como se muestra en la siguiente figura, si en alguna de las unidades primarias (cuadrados azules) se cumplió con el criterio C entonces las zonas adyacentes entrarían a la muestra (cuadrados rojos). Esto permite, enfocar más recursos en aquellas zonas que tienen el criterio de interés.

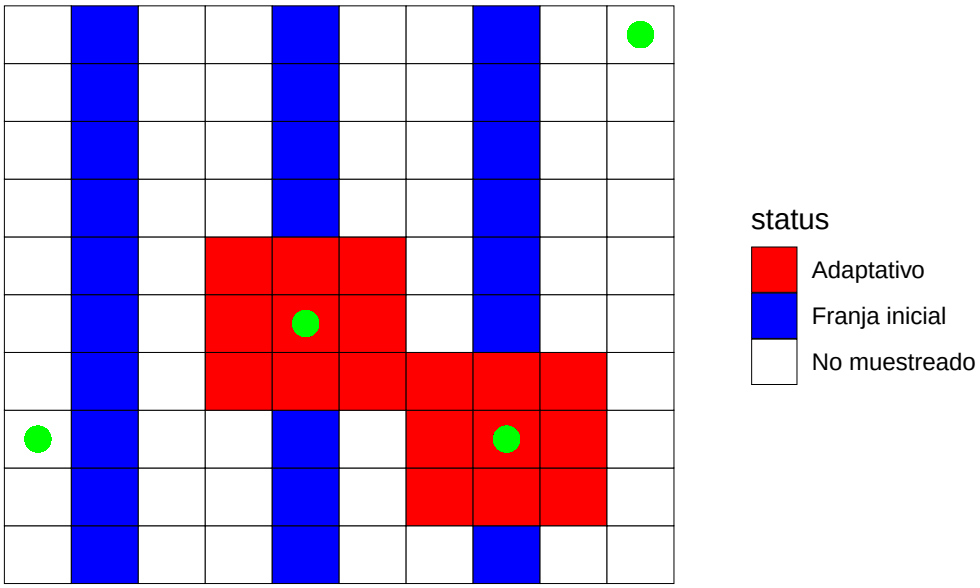

Figura 9: Proceso adaptativo de muestreo en franjas

Por otra parte, también se puede definir otra regla de selección para las unidades primarias, por ejemplo que entren en muestra las filas pares, pero tomando en muestra un cuadrante y el que sigue no; tal como se muestra en la siguiente figura. Esa es la principal diferencia con el método de franjas, en este caso la regla de asignación para las unidades primarias puede ser de otro tipo.

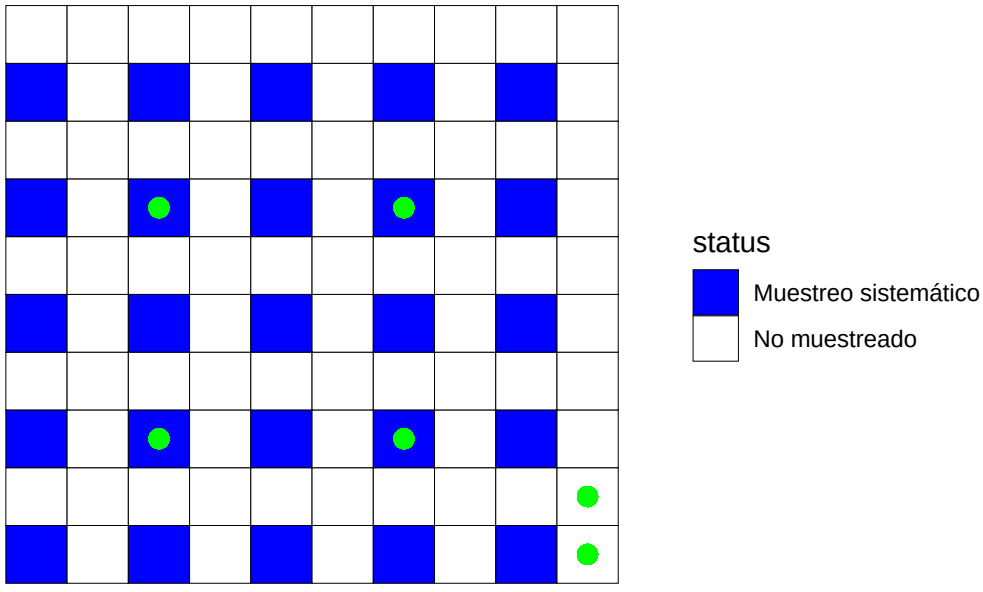

Figura 10: Muestreo inicial en sistemático

Después de haber muestreado las unidades primarias, si en alguna de ellas se cumplió con el criterio C (puntos verdes), entonces entra en juego el proceso adaptativo que implica muestrear a las zonas vecinas, tal como se muestra en la siguiente imagen por los cuadrados rojos. Estos métodos implican

una forma de selección inicial diferente de la de muestreo aleatorio simple que se aplica en el muestreo adpatativo de conglomerados.

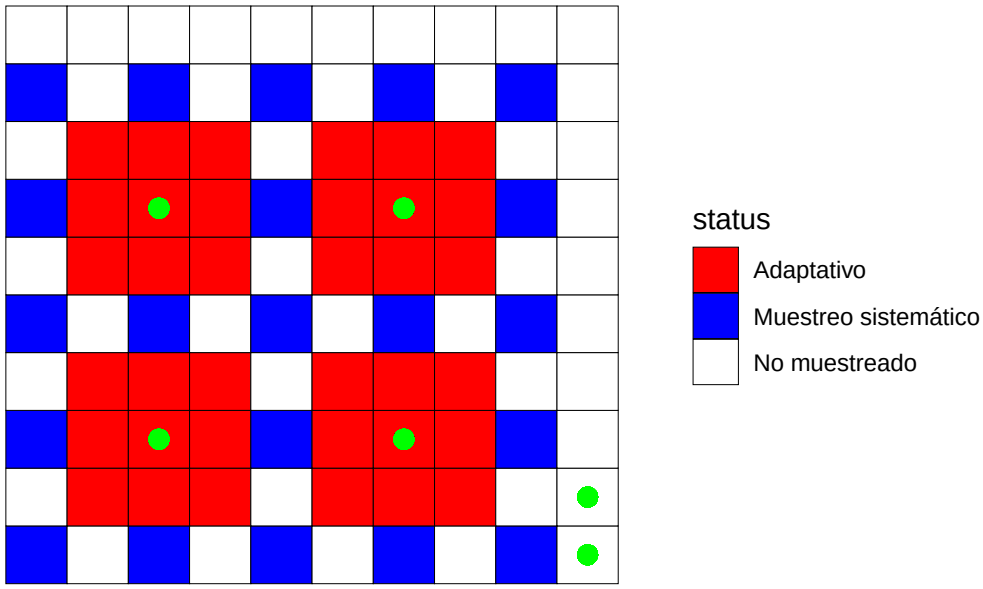

Figura 11: Proceso adaptativo en muestreo sistemático

Cuando las unidades primarias consisten de conjuntos de unidades que se encuentran distribuidas de manera uniforme sobre la población, de acuerdo con algún orden, entonces se denominará muestra inicial sistemática. Y cuando las unidades primarias consistan de filas o columnas de unidades, entonces se denominarán muestra inicial de franjas.

2.6 Muestreo estratificado adaptativo por conglomerados

Por último, para el muestreo estratificado, al igual que en el procedimiento tradicional, la población se divide en estratos (grupos), de los cuales se toma una muestra y se mide la variable de interés; la diferencia recae en que para el muestreo adaptativo si la unidad que se muestrea cumple con la condición C especificada, entonces se agregan a la muestra las unidades vecinas, tal como ya se ha descrito en las anteriores dos variantes. Este diseño tiene la complicación de que, al hacerse en un territorio dividido por estratos, hacer selección en un estrato puede dar lugar a que las unidades que se agregan como consecuencia del proceso adaptativo pertenezcan a otro estrato. Pero este diseño aprovecha tanto la información adicional que brinda de manera *a priori* el posible conocimiento de estratos y el proceso adaptativo que permite explotar a las muestras que brinden más información.

3 Referencias

Box, J.B., R.M. Dorazio, y W.D. Liddell. (2002). Relaciones entre las características del sustrato del lecho fluvial y mejillones de agua dulce (Bivalvia: Unionidae) en arroyos de llanura costera. Revista de la Sociedad Bentológica Norteamericana 21:253-260

O'Gorman, T. W. (2004). *Applied adaptive statistical methods: tests of significance and confidence intervals*. Society for Industrial and Applied Mathematics.

Pereyra L., & Vaira, M. (2021). *El diseño de muestro.* Recuperado de https://www.researchgate.net/publication/356565224_EL_DISENO_DE_MUESTREO

Roesch, F.A., Jr. (1993). Muestreo adaptativo por conglomerados para inventarios forestales. Ciencia Forestal 39:655-669.

Seber, G., & Salehi, M. (2012). *Adaptive sampling designs: inference for sparse and clustered populations*. Springer Science & Business Media.

Thompson, S. K. (2012). *Sampling* (Vol. 755). John Wiley & Sons.

Thompson, S.K., y F.L. Ramsey. (1983). Muestreo adaptativo de poblaciones animales. Informe Técnico 82. Departamento de Estadística, Universidad Estatal de Oregón, Corvallis, OR.

Vesely, D.G. y J. Stamp. (2001). Reporte de encuestas para anfibios de riachuelos realizados en el Bosque Estatal Elliott, junio-septiembre de 2001. Informe elaborado para el Departamento Forestal de Oregón. Pacific Wildlife Research, Corvallis, OR.

4 Anexo: Código para las Simulaciones

4.1 Código: Ejemplo 1.

La simulación del ejemplo 1 se realizó en R y este fue el código empleado.

```
# Paquete necesario para la visualización
library(ggplot2)

# Comenzamos simulando un bosque de 100 parcelas
# Fijamos una semilla
set.seed(42)

# Fijamos el número de parcelas y tamaño de la cuadrícula
n_unidades <- 100
n_filas <- 10
n_columnas <- 10

# Se define una probabilidad (baja) de que una unidad tenga un árbol de interés con alguna
# característica rara o difícil de encontrar, se define una baja para que sean pocas las
# parcelas con ests características.

prob_raro <- 0.05

# Llamaremos a estos árboles como "raros"
# Por tanto simularemos si un árbol es raro con: 1 = raro, 0 = no raro
bosque <- sample(c(0, 1), size = n_unidades, replace = TRUE, prob = c(1 - prob_raro, prob_raro))

# Se representará el bosque con una matriz 10x10
bosque_matrix <- matrix(bosque, nrow = n_filas, ncol = n_columnas)
bosque_matrix

# Definir la función para expandir el muestreo
expandir_muestreo <- function(muestras, bosque_matrix, n_filas, n_columnas) {
  muestras_ampliadas <- muestras
  for (i in muestras) {
    fila <- (i - 1) %/% n_columnas + 1
    columna <- (i - 1) %% n_columnas + 1
    if (bosque_matrix[fila, columna] == 1) {  # Si encontramos un árbol raro
      # Entonces se agregarán vecinos
      vecinos <- c(i - 1, i + 1, i - n_columnas, i + n_columnas)  # En todas las direcciones: Arriba, abajo, izquierda, derecha
      vecinos <- vecinos[vecinos > 0 & vecinos <= n_unidades]  # Filtrar valores fuera del rango
      muestras_ampliadas <- unique(c(muestras_ampliadas, vecinos))
    }
  }
  return(muestras_ampliadas)
}

# Se seleccionan inicialmente muestras aleatorias
muestras <- sample(1:n_unidades, size = 5)
muestras_ampliadas <- muestras

# Expandemos el muestreo adaptativo
muestras_ampliadas <- expandir_muestreo(muestras_ampliadas, bosque_matrix, n_filas, n_columnas)

# Crear un data.frame para la visualización
bosque_df <- expand.grid(x = 1:n_columnas, y = 1:n_filas)
bosque_df$raro <- as.vector(bosque_matrix)
bosque_df$muestreado <- ifelse(1:n_unidades %in% muestras_ampliadas, 1, 0)

# Visualización mejorada del proceso de muestreo adaptativo
ggplot(bosque_df, aes(x = y, y = x)) +
    geom_tile(aes(fill = factor(raro)), color = "grey80") +
    geom_point(data = subset(bosque_df, muestreado == 1),
               aes(x = x, y = y),
               shape = 21, color = "red", fill = "red", size = 3) +
    scale_fill_manual(
        values = c("white", "forestgreen"),
        labels = c("No raro", "Raro"),
        name = "Tipo de árbol"
    ) +
    labs(
        title = "Proceso de Muestreo Adaptativo en Bosque",
        subtitle = "Puntos rojos indican parcelas muestreadas",
        x = "",
        y = ""
    ) +
    theme_minimal(base_size = 14) +
    theme(
        axis.text = element_blank(),
        axis.ticks = element_blank(),
        panel.grid = element_blank(),
        legend.position = "top"
)
```

4.2 Código: Ejemplo 3.

El código de la simulación del ejemplo 3 se realizó usando el lenguaje de programación `Julia`.

```
#Cargamos paqueterias
using Plots, Statistics,CSV, DataFrames
using LinearAlgebra
using ColorSchemes

#dado un  elementos quiero conocer sus vecinos, Matriz y coordenada de la unidad
function vecindad(M,ui)
    n,m= size(M)
    vecino(k,a)= mod1(k,a) #vecinos con periocidad

    i= ui[1]
    j= ui[2]

    vecinos= [  M[vecino(i-1,n), j],
                M[vecino(i+1,n), j],
                M[i, vecino(j-1,m)],
                M[i, vecino(j+1,m)]]

    coordenadas= [ (vecino(i-1,n), j),
                    (vecino(i+1,n), j),
                    (i, vecino(j-1,m)),
                    (i, vecino(j+1,m))]

    return vecinos, coordenadas
end

# La función que crecerá según un valor C, me arroja coordenas en la matriz

function crece(U, s_i, C)

visitados= Set()
cola= [s_i]
Ci=[s_i]

while !isempty(cola)
    celda = popfirst!(cola)

    if celda in visitados
        continue
    end

    push!(visitados, celda)

    x,y = celda
    if U[x,y] > C
        push!(Ci, celda)

        vecinos = vecindad(U, celda)
        for i in 1:4
            if vecinos[2][i] no visitados
                push!(cola, vecinos[2][i])
            end
        end

    end
end
return unique(Ci)

end

function redes(U, mas, C)
    l = length(mas)
    mi =  []
    y = []
    ni= []


    for i in 1:l
        r= crece(U,mas[i],C)
        n= length(r)
        mi= push!(mi,n)
        yi= [U[r[j][1],r[j][2]] for j in 1:n]
        y= push!(y,yi)
        ni= push!(ni, r)
    end

    return  mi,  y, ni
end

#Cargamos datos y graficamos
data = DataFrame(CSV.File("data.csv"))

scatter(data.x,data.y, ratio= 1, color= "purple", label= "", title= "Datos")
```

```
#una matriz para almacenar las cuentas
U = zeros(Int, 20, 20) #unidades

x_i = floor.(Int, data.x)
y_i = floor.(Int, data.y)

for (i, j) in zip(x_i, y_i)
    if 1 <= i <= 20 && 1 <= j <= 20
        U[i, j] += 1
    end
end

mas= [(rand(1:20), rand(1:20)) for i in 1:13];  #13 unidades
muestra = redes(U, mas, 3)

function resaltar_zona(zona)
    xs = [x for (x, y) in zona]
    ys = [y for (x, y) in zona]
    scatter!(ys, xs, label= "")
end

mapa_calor

resaltar_zona.(muestra[3])
plot!(title= "Crecimiento de u_i")

wi= [(1/muestra[1][i]) * sum(muestra[2][i]) for i in 1:13]

resultados= DataFrame(
    m_i = muestra[1],
    y_i = muestra[2],
    w_i = wi
)

N= 20*20
n= sum(resultados.m_i) #tamaño de muestra
media_est= 1/n * sum(resultados.w_i)
varianza_est= (N-n)/(N*n*(n-1)) * sum((resultados.w_i .- media_est).^2)

println("La media estimada= $media_est")
print("Con varianza=$varianza_est")
```

4.3 Código: Ejemplo 4.

Para la simulación del ejercicio 4 se uso el lenguaje de programación `Python 3.12.4`.

```
import networkx as nx
import matplotlib.pyplot as plt
import random
import numpy as np
from matplotlib.colors import Normalize
from matplotlib.cm import ScalarMappable
import matplotlib
import pandas as pd
matplotlib.rcParams['figure.figsize'] = 8, 10

def generar_vecindades(familia, max_vecinos, seed=1):
    np.random.seed(seed)
    random.seed(seed)  # Configurar la semilla para random
    for id_mun in familia:
        num_vecinos = np.random.binomial(n=max_vecinos, p = 0.65, size=1).item()

        vecinos = random.sample([m for m in familia if m != id_mun], num_vecinos)
        familia[id_mun]["vecinos"] = vecinos

        # Aseguramos que las conexiones sean bidireccionales
        for vecino in vecinos:
            if id_mun not in familia[vecino]["vecinos"]:
                familia[vecino]["vecinos"].append(id_mun)

    return familia


def simular_datos(num_familias, max_miembros, max_vecinos, seed=1):
    familia = {}
    np.random.seed(seed)
    random.seed(seed)  # Configurar la semilla para random
    for i in range(1, num_familias + 1):
        cases = np.random.poisson(0.35)
        familia[i] = {
            "miembros": np.random.choice(range(2, max_miembros), 1).item(),
            "vecinos": []
        }
```

```
        familia[i]["casos"] = cases if cases <= familia[i]["miembros"] else familia[i]["miembros"]

    new_familia = generar_vecindades(familia, max_vecinos, seed)
    return new_familia


def muestreo_adaptativo(familia, tamano_inicial, C_min, seed=1):
    np.random.seed(seed)
    random.seed(seed)  # Configurar la semilla para random
    mas = np.random.choice(list(familia.keys()), tamano_inicial, replace=False)
    muestra = mas.tolist()
    evaluados = set()
    recorrido = []  # Para guardar el orden del recorrido

    while True:
        nuevos_familia = []
        for mun_id in muestra:
            if mun_id not in evaluados:
                evaluados.add(mun_id)
                recorrido.append(mun_id)
                if familia[mun_id]["casos"] >= C_min:
                    vecinos = familia[mun_id]["vecinos"]
                    for vecino_id in vecinos:
                        if vecino_id not in muestra and vecino_id not in nuevos_familia:
                            nuevos_familia.append(vecino_id)

        muestra.extend(nuevos_familia)

        if not nuevos_familia:
            break

    return muestra, recorrido

# 3. Graficar con contornos y aristas personalizadas
def graficar_muestra(familia, muestra, recorrido):

    G = nx.Graph()

    # Crear nodos y aristas
    for mun_id, data in familia.items():
        G.add_node(mun_id, casos=data["casos"])
        for vecino in data["vecinos"]:
            G.add_edge(mun_id, vecino)

    # Obtener la intensidad del color basada en los casos
    casos_maximos = max([familia[n]["casos"] for n in G.nodes()])
    normalizador = Normalize(vmin=0, vmax=casos_maximos)  # Normalizar valores de casos

    # Crear lista de colores para nodos
    colores = [normalizador(familia[n]["casos"]) for n in G.nodes()]
    bordes = ['black' if n not in muestra else 'red' for n in G.nodes()]  # Contorno azul para seleccionados
    node_width = [1 if n not in muestra else 3 for n in G.nodes()]

    # Identificar aristas conectadas a nodos seleccionados
    edge_colors = []
    widths = []
    for u, v in G.edges():
        if u in muestra and v in muestra:
            edge_colors.append('green')  # Aristas entre seleccionados
            widths.append(3)
        else:
            edge_colors.append('gray')  # Aristas no seleccionadas
            widths.append(1)

    # Graficar el grafo
    pos = nx.spring_layout(G, seed=42)  # Layout del grafo
    plt.figure(figsize=(10, 8))
    plt.set_cmap("Purples")
    nx.draw_networkx_nodes(G, pos, node_color=colores, edgecolors=bordes, node_size=500, alpha=0.9, linewidths=node_width)
    nx.draw_networkx_edges(G, pos, edge_color=edge_colors, alpha=0.6, width=widths)
    nx.draw_networkx_labels(G, pos, font_size=8, font_color="black")

    # Crear el mappable para la barra de color
    sm = ScalarMappable(norm=normalizador)
    sm.set_array([])
    cbar = plt.colorbar(sm, ax=plt.gca(), shrink=0.8)
    cbar.set_label("Número de casos", fontsize=12)

    # Agregar título
    plt.title("", fontsize=1)
    plt.axis('off')
    plt.tight_layout()
    plt.savefig("img/ejem4_disease.jpg", bbox_inches='tight', dpi=400)
    plt.show()

def get_conections(nodos, id):
    # Crear el grafo
    G = nx.Graph()

    for mun_id, data in nodos.items():
```

```
        G.add_node(mun_id)
        for vecino in data["vecinos"]:
            G.add_edge(mun_id, vecino)

    # Identificar componentes conexas
    vecinos_distancia_1_todos = {nodo: list(G.neighbors(nodo)) for nodo in G.nodes()}
    vecinos_distancia_1_todos = {k:v for k,v in vecinos_distancia_1_todos.items() if k in id}


    return vecinos_distancia_1_todos

def bfs(nodo, visitados, diccionario):
    subred = []
    cola = [nodo]
    while cola:
        actual = cola.pop(0)
        if actual not in visitados:
            visitados.add(actual)
            subred.append(actual)
            cola.extend(diccionario.get(actual, []))
    return subred

def generar_dataframe_subredes(diccionario):

    visitados = set()
    subredes = []
    nodos_llave_por_subred = []

    for nodo in diccionario:
        if nodo not in visitados:
            subred = bfs(nodo, visitados, diccionario)
            subredes.append(subred)
            # Nodos que son llaves dentro de esta subred
            nodos_llave = [n for n in subred if n in diccionario]
            nodos_llave_por_subred.append(nodos_llave)

    # Crear el DataFrame
    data = {
        "SubRed": list(range(0, len(subredes))),
        "#Nodos": [len(subred) for subred in subredes],
        "Nodos": [", ".join(map(str, sorted(subred))) for subred in subredes],
        "Nodos en Muestra": [", ".join(map(str, sorted(llaves))) for llaves in nodos_llave_por_subred],  # Nodos que son llaves
    }
    df = pd.DataFrame(data)
    return df

# Ejecución
Nfamilia = 100
SEMILLA = 423204
NINICIAL = 10
CRITERIO = 1
MAXPOP = 12
MAXVECINOS = 4

datos_familia = simular_datos(Nfamilia, MAXPOP, MAXVECINOS, SEMILLA)
muestra, recorrido = muestreo_adaptativo(datos_familia, NINICIAL, CRITERIO, SEMILLA)

print(f"Muestra: {muestra},\nn = {len(muestra)}")

muestra_nodos = {mun:datos_familia[mun] for mun in muestra}

connections = get_conections(datos_familia, muestra)

# Subredes
df_subredes = generar_dataframe_subredes(connections)
print(df_subredes.head())

#graficar_muestra(datos_familia, muestra, recorrido)
```