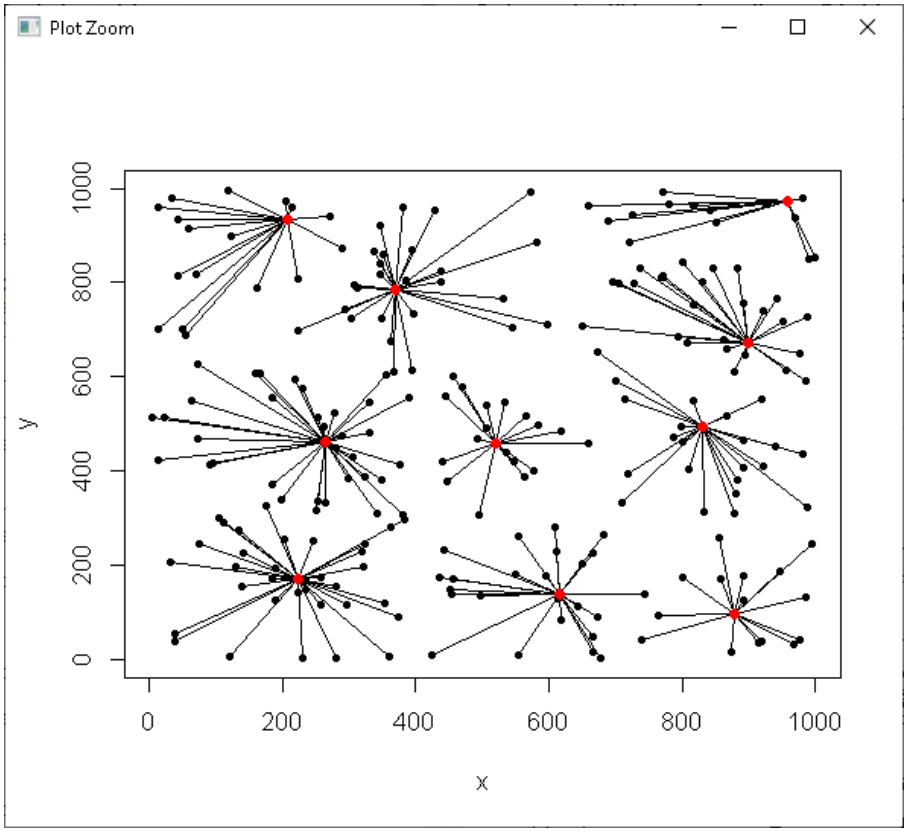
Algoritmo: Búsqueda Aleatoria Localizada

Problema: Segundo

El problema en cuestión busca asignar un punto del espacio a algún centro, esto con el fin de minimizar la suma de distancias entre cada punto con su respectivo centro.

Para este problema, nos será posible apreciar (también acompañados de la intuición) que se espera ver centros conteniendo a sus puntos cercanos, de esa forma se minimizarán las distancias. Ya que si tuviéramos centros que tocasen puntos al otro lado del mapeo, nos alejaría de una solución óptima. Dicho eso, una posible solución al problema es la siguiente, donde es claro cómo cada punto tiende a su centro más cercano.



Para hacer posible un algoritmo que encuentre soluciones cercanas a la óptima en referencia a minimizar distancias, es necesario definir un modelo que cuente con ciertas características, las cuales son:

- **Representación de una solución.** Para resolver este primer reto, se utiliza un vector donde cada casilla va a hacer las veces de un punto, similar a como se vería la implementación de una cubeta. Y a su vez, esa casilla contiene un número que será el identificador del centro al que va a apuntar nuestro punto en el espacio.
- **La evaluación de la función objetivo.** La función que se va a encargar de decirnos qué tan buena es nuestra solución, será la suma del valor que otorgue cada centro, siendo este, la adición de las distancias con sus puntos asignados del espacio. Por muy complicada que pueda parecer esta implementación, en la realidad es que se conforma apenas por una variable que guarde la suma mientras se itera una vez el vector solución.
- **Mecanismo de perturbación.** El reto que implicaba más creatividad de este trabajo fue diseñar un mecanismo de perturbación que nos acercase a la solución óptima. La idea que se implementó al final consistió en generar un vector pseudoaleatorio que asignara el rol de centro a ciertos puntos, no es completamente aleatorio porque este debe cumplir el requerimiento de tener cada centro a una distancia mayor a 175, de esta forma se fomenta una buena dispersión en el espacio. El siguiente paso es trabajar sobre ese vector para asignar un centro aleatorio a cada punto y con el paso de las iteraciones, se buscará tomar un conjunto de vértices los cuáles serán optimizados llevándolos a su centro definitivamente más cercano. Aunque esta solución tiene aires de fuerza bruta, la realidad es que es una metaheurística escalable, esto es, es posible que funcione con entradas aún mayores y tendrá un buen comportamiento, aunque evidentemente el número de iteraciones para una solución decente será directamente proporcional al tamaño de la entrada.

Conclusiones.

La implementación de estas ideas resultó en un algoritmo capaz de encontrar soluciones cuya representación gráfica es acertada, y aunque en algunos casos es posible mejorar el resultado, la verdad es que los indicadores estadísticos se verían mayormente afectados al cambiar el mecanismo de perturbación, el cuál es crucial para encontrar una buena solución.

Al final, tras correr 50 veces el mismo algoritmo de forma independiente, se obtuvieron los siguientes indicadores estadísticos.

```
> min(soluciones)
[1] 31747.37

> max(soluciones)
[1] 47211.44

> sd(soluciones)
[1] 2616.618

> mean(soluciones)
[1] 36496.6
```

Los cuales representan a un algoritmo con soluciones poco estables que rondan los 30,000 y 50,000 puntos respecto a la función objetivo.

En caso de necesitar mejores soluciones, sería buena idea explorar distintos algoritmos, o hacer pruebas con otros mecanismos de perturbación, ya que, para este caso en específico, un mayor número de iteraciones no representaría gran cambio pasado cierto límite.