UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE INFORMÁTICA

DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Trabajo Fin de Grado en Ingeniería Informática

Implementación de un nuevo algoritmo de clasificación en el lenguaje R.
Implementation of a new classification algorithm in the R language.

Dirigido por:

Sonia Estévez Martín

John Erik Ibarra Guerrón

Curso académico 2021-22

Convocatoria Junio

Acknowledgements

I want to thank my project director, Sonia Estévez Martín and Victoria López for all the support, guidelines and advice they have given me throughout the development of this project, thanks to their help, it has been carried out in the best possible way.

On the other hand, I also want to thank my parents for all the effort they have made to give me the opportunity to have a good education. my girlfriend and my brother for all their moral support and encouragement that they have given me throughout these years.

Abstract

Nowadays the improvements in urban infrastructure that has been made in Madrid Central, along with the new regulations for closing to road traffic has led to the influx of walkers, mainly in the summer months, which causes a number of problems in those areas where agglomerations are not expected because they are not structurally equipped for it.

Therefore, we propose the implementation of a new classification algorithm based on the fusion of K-MEANS and Hierarchical clustering that allows us to know which streets share a similar average number of pedestrians in the holiday period, besides in this way we can apply urban improvements based on the classification generated in those areas with the highest number of walkers.

Keywords

Algorithm, Analytics, Machine Learning, Kmeans, Hierarchical Clustering, Cluster, Dataset, Dataframe, Data Science, R.

Resumen

En los últimos años las mejoras en infraestructura urbana que se ha realizado en Madrid Central, junto con la nueva normativa de cierre al tráfico rodado ha propiciado la afluencia de peatones, principalmente en lo meses de verano, lo que ocasiona una serie de problemas en aquellas zonas donde no se prevé aglomeraciones debido a que no están dotadas estructuralmente para ello.

Por ende se propone la implementación de un nuevo algoritmo de clasificación basado en la fusión de K-MEANS y Hierarchical clustering que nos permita conocer que calles comparten un promedio de peatones similar en el periodo vacacional, así, de este modo se puede aplicar mejoras urbanas basándonos en la clasificación generada en aquellas zonas con mayor número de peatones.

Palabras Clave

Algoritmo, Análisis, Aprendizaje Automático, Kmeans, Clúster Jerárquico, Cluster, Conjunto de datos, Dataframe, Ciencia de datos, R.

Índice general

Índice de figuras

1. Introducción

En la actualidad, el uso de herramientas de Big Data y Deep Learning ha aumentado considerablemente lo que ha beneficiado a la obtención, almacenamiento y procesamiento de grandes volúmenes de datos que posteriormente son usados por las empresas y organizaciones de ámbito publico y privado para la toma de decisiones, por ende, es necesario crear nuevos algoritmos que sirvan de ayuda a la hora de transformar la información de los conjuntos masivos de datos en valor.

En nuestra vida diaria se aplican este tipo de soluciones en diversas áreas desde entidades bancarias, telecomunicación, salud, militar, investigaciones tecnologías etc... Todos estos escenarios tienen algo en común, una cantidad enorme de información que se debe procesar y que años atrás, suponían un trabajo muy costoso a nivel computacional, la aparición de la infraestructura cloud, la mejora de los procesadores y la aparición de algoritmos cada vez más rápidos y eficaces está ayudando a la sociedad a alcanzar un mayor avance tecnológico.

Pese a que existen algoritmos que han sido ampliamente utilizados, contrastados y que generan buenos resultados, hay casos en los que a pesar de generar clasificaciones robustas, no son de un valor significativo ya que pueden crear subgrupos muy numerosos pero de escasos individuos o por el contrario pocos grupos pero de gran tamaño, lo que no resulta relevante, por ello los profesionales del sector siguen buscando y creando nuevos algoritmos para enfrentarse a los nuevos problemas y que son de diferentes tipos y complejidad:

- **Predicción**: se realizan mediante modelos entrenados.
- **Clasificación**: se trata de problemas en los que se busca encontrar

similitudes entre elementos de una población de estudio, analizando variables relevantes se llega a subgrupos con elementos afines entre sí.

- **Aprendizaje supervisado**: se trata de problemas en los que se busca inferir conocimiento en un sistema mediante pares de entrada salida, de está manera se pretende aprender mediante datos y modelos precargados y de esta manera inferir conocimiento mediante deducción.
- **Aprendizaje no supervisado**: es un tipo de aprendizaje que no requiere de pares entrada salida, de esta manera partimos de datos sin etiquetar y es el algoritmo el que se encarga de intentar entender los datos y extraer información relevante.

En este proyecto se trabaja sobre un problema de agrupamiento a partir de un conjunto de datos donde la clasificación se infiere a partir de similitudes, se trata de determinar la semejanza entre elementos basándonos en una serie de variables que consideramos relevantes.

1.1. Motivación

El nuevo algoritmo que se plantea en este proyecto pretende ayudar a mejorar el transito de los peatones en el mes de mayor afluencia en Madrid Central, la zona más concurrida de la capital a las 20h, de esta forma podemos ofrecer una serie de subconjuntos con las calles que tienen similitud de promedio de peatones y así poder saber cuales son las que necesitan adaptar su infraestructura para evitar aglomeraciones y mejorar su transito, para ello utilizaremos dos de los algoritmos más utilizados en la ciencia de datos K-Means y Hierarchical Clustering, para la creación del algoritmo Merge que se basa en la fusión de ambos algoritmos para generar una clasificación determinada por el tamaño de los clusters.

1.2. Objetivos

La implementación del algoritmo Merge planteado en [4] mediante el lenguaje de programación R [5] en RStudio [6], dónde se quiere mejorar los resultados generados por dos de los algoritmos más usados dentro de la ciencia de datos para el tratamiento de información K-Means y Hierarchical clustering

realizando una fusión de ambos para generar el número de conjuntos óptimo con una cantidad de individuos equilibrada aplicado a nuestro conjunto de datos de Madrid Central al que se puede acceder desde ***Aforo peatones madrid*** del portal de datos abiertos de Madrid [7].

1.3. Plan de trabajo

Una vez presentada la finalidad del desarrollo del proyecto y los objetivos que se persiguen voy a describir el plan de trabajo que se ha seguido.

Definimos junto con la directora de proyecto cómo se realizaría el seguimiento del desarrollo del TFG, acordando realizarse reuniones periódicas en las que se comentan los avances e ideas que iban surgiendo, recibiendo por su parte correcciones o propuestas de mejora que se pondrían aplicar y que se revisaría su implementación en las sesiones posteriores. A continuación, se detalla el desarrollo del trabajo realizado.

Como primer paso, se ha recopilado información sobre el funcionamiento de los algoritmos de clustering K-Means y Hierarchical Clustering que se emplean en este proyecto en la elaboración del nuestro algoritmo Merge, de esta manera nos familiarizamos con su uso y aplicación en R.

Con los conceptos claros de ambos algoritmos se realizó una búsqueda de un conjunto de datos que nos permitió poner a prueba el nuevo algoritmo dicho dataset se puede conseguir de los datos públicos que ofrece la Comunidad de Madrid en la web **https://datos.madrid.es/portal/site/egob**.

Posteriormente se ha procedido a la implementación del algoritmo mediante el lenguaje de programación R con la herramienta Rstudio, se ha realizado un análisis de la clasificación generada, recopilado gráficas con datos de interés y modificaciones pertinentes en base al análisis previo.

El código de la implementación de este proyecto se encuentra a disposición en el siguiente repositorio público **https://github.com/johnIbarra28/TFG-2021-2022**

1.4. Antecedentes

Los algoritmos de clustering [8] tienen como principal objetivo encontrar y agrupar lo que se conoce como clúster, elementos de un conjunto de datos que tienen características similares, que a su vez tienen propiedades diferentes a las de los individuos de otra agrupación.

Este tipo de proceso se utiliza en machine learning en el entrenamiento de modelos de tipo no supervisado y que se usan en diversos ámbitos de la ciencia de datos

1.5. Tipos de clasificación

- **Duro**: El clustering duro es el agrupamiento donde cada uno de los elementos que componen un clúster pertenece exclusivamente a dicho conjunto.
- **Blando**: El clustering suave es el agrupamiento de los elementos donde no restringe la pertenencia a un único clúster, sino que su salida es una probabilidad de pertenecer a cada uno de los grupos que se crean.

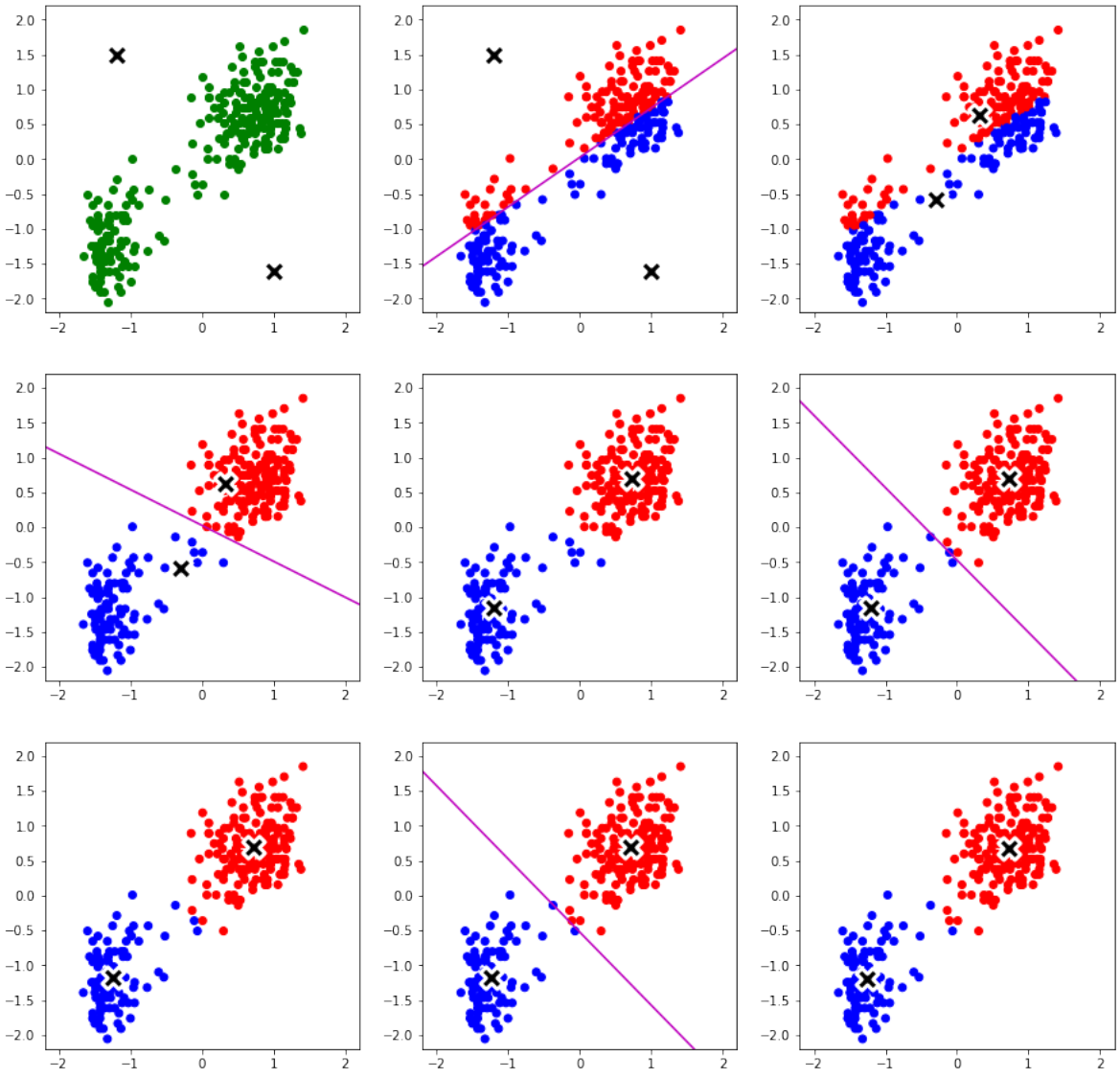

Figura 1.1: Imagen de clustering tomada del internet del trabajo [1]

2. Estados del arte

2.1. Algoritmos de clustering

2.1.1. K-Means

El algoritmo K-Means [9][10] realiza una clasificación no supervisada para agrupar elementos con características similares en K grupos dónde la k es establecida por el usuario, se utiliza la distancia mínima entre los elementos y el centroide del grupo generado. Existen varias distancias que puede usar el algoritmo para realizar su propósito, siendo la más común la distancia euclídea, este algoritmo tiene como característica importante una buena escalabilidad en función del tamaño del conjunto de datos al que se vaya a aplicar.

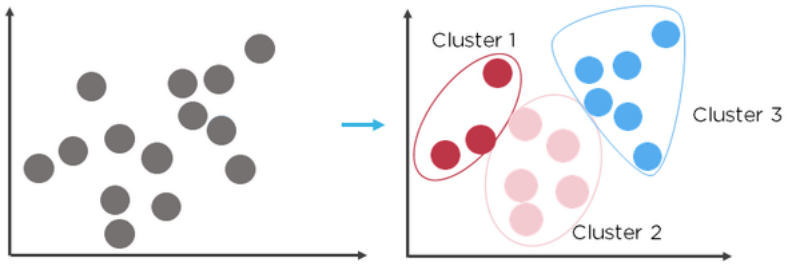

Figura 2.1: K-Means imagen obtenida de la web [2]

El algoritmo consta de tres pasos:

- **Inicialización**: En esta fase se establece el número de k conjuntos que se va a crear y la posición de los k centroides de forma aleatoria.
- **Asignación**: Se asigna cada elemento a su centroide más próximo.

- **Actualización**: Se establece como nueva posición de los centroides, la media aritmética de las posiciones de los elementos asignados al grupo.

Estos tres pasos se repiten hasta que no se puedan seguir actualizando las posiciones de los centroides como se puede observar en la Figura [1].

2.1.2. Hierarchical clustering

El algoritmo de clustering jerárquico se basa en la creación sucesiva de agrupaciones que se van generando por si solas en cada iteración, buscando el número óptimo de clusters a diferencia de algoritmos no jerárquicos en los que se debe definir el numero de grupos.

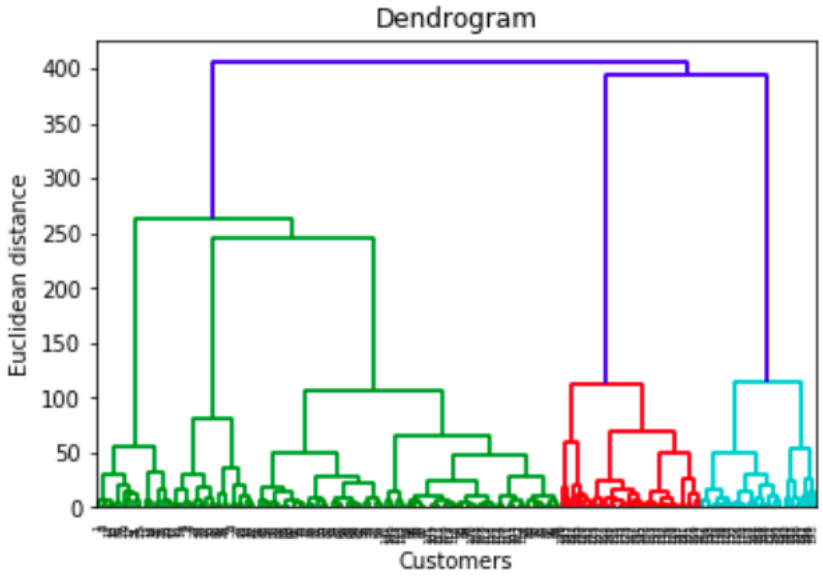

Figura 2.2: Hierarchical clustering imagen obtenida de la web [3]

Este algoritmo tiene dos posibles estrategias de aplicación:

- **Aglomerativas**: Se parte de varios elementos y se van juntando aquellos más similares formando grupos de clusters óptimos.
- **Disociativas**: Se parte un grupo grande de elementos que se irán separando en función de sus diferencias hasta quedarse grupos de clusters óptimos.

Los algoritmos jerárquicos tienen diferentes métodos de agrupación: single linkage, complete linkage clustering, unweighted pair-group arithmetic averages, minimum variante clustering , centroide , mediana.

2.2. Definición del nuevo algoritmo Merge

2.2.1. Nomenclatura

Partimos de los clúster generados por K-Means y Hierarchical clustering de este modo podemos denotar ambos resultados de la siguiente manera:

- Agrupaciones generadas por K-Means:

  $$k$$

  Tamaño de la partición del conjunto de observaciones.

  $$C = \{C_1, ..., C_k\}$$

- Agrupaciones generadas por Hierarchical clustering:

  D_α =una partición del conjunto de observaciones y $\alpha - cut$ es el nivel de profundidad dentro del dendograma.

  $$D_{\alpha-cut} = \{D_1, ..., D_m\}$$

- Notación dendograma

  $$A = \{r, l\}$$

  - A* conjunto de letras del alfabeto.
  - 2 hijo derecho dentro del dendograma.
  - 1 hijo izquierdo dentro del dendograma.
  - O observacion que contiene una secuencia de letras del alfabeto indicando el camino dentro de un arbol binario.

"Dos observaciones O_1, O_2, pertenecen al mismo clúster D_i si y solo si ambas tienen el prefijo común $\alpha = 1 + \#connections(a - cut)$".

Tras definir la nomenclatura que vamos a usar podemos decir que C y Da corresponden a los clusters generados por K-mean y Clustering jerárquico, a continuación, se detalla los pasos de la implementación del nuevo algoritmo.

2.2.2. Implementación

- Paso 1

  Fijamos el tamaño N de clusters que vamos a generar con nuestro algoritmo Merge y procedemos a obtener los tamaños del camino desde la raíz hasta la hoja para cada una de las observaciones dentro del dendograma.

- Paso 2

  A continuación; ordenamos de manera descendente en función de los tamaños obtenidos en el paso 1.

- Paso 3

  Definimos la condición de cruce de K-Means y de Herarchical clustering:

$$\forall i \in \{1...k\}\ \forall j \in \{i...k\} :\ \texttt{if}\ \exists o_1 \in C_i, \exists o_2 \in C_j :$$
$$(|o_1| = |o_2| = n \vee |o_1| = |o_2| - 1 = n) \wedge o_1[1 : n - p] = o_2[1 : n - p]$$

  De esta forma ambas observaciones tienen el mismo prefijo, y por tanto procedemos a cruzarlos.

  Y a continuación actualizamos:

$$\texttt{MR} = \texttt{MR} - \{C_i, C_j\} \cup \{C_{ij}\}$$

- Paso 4: fijamos $p = p + 1$

- Paso 5: Si $p < N$ volvemos al paso 3, sino continuamos con el paso 6.
- Paso 6: Agrupamos en un único cluster aquellas observaciones no agrupadas.
- Paso 7: Comprobamos el tamaño de los nuevos clusters, en caso de que su longitud sea superior a la fijada por el usuario en el paso 1, procedemos a dividirlos en clusters de tamaño N.

3. Conjunto de datos

El conjunto de datos que se utilizado en el proyecto ha sido obtenido de la web datos abiertos [7], en la que el ayuntamiento de Madrid pone a disposición de la ciudadanía y en la que se ofrece un catálogo muy completo de información que se ha ido recopilando a lo largo de los años por parte del ayuntamiento, los datasets se pueden descargar en diferentes formatos según se requieran (RDF, CSV, EXCEL, AVRO, JSON, XML, GEO).

El dataset contiene información relativa a la cantidad de peatones en determinadas calles del centro de la capital de España en varias fechas y horas a lo largo los años y se actualiza cada trimestre, en este caso para la implementación de nuestro nuevo algoritmo, optamos por utilizar el archivo que comprende el periodo desde el 1 de enero al 31 de diciembre de 2020, ya que la información recogida nos ofrecía datos interesantes debido a la pandemia. A continuación, se analizará el estado inicial del dataset.

Como se puede observar en las Figuras 3.1 y 3.2 nuestro conjunto contiene doce columnas que refleja una serie de atributos recogidos por sensores distribuidos en la zona centro de Madrid y más de 166000 registros generados a lo largo del año.

- **Fecha**: contiene el día, mes, año y hora en la que se realizó la toma de los datos.
- **Hora**: en la que se ha realizado la toma de los datos, como se puede observar el intervalo entre cada medición se produce cada hora.
- **Identificador**: en función de las calles en las que se realiza la medición.
- **Peatones**: cantidad de peatones en una determinada fecha, hora, dirección en general, latitud y longitud.

- **Número de distrito**: distrito en el que se realiza la medición.
- **Nombre vial**: nombre de la calle en la que toman los datos.
- **Número**: número de la calle.
- **Código postal**: código postal del distrito
- **Observación dirección**: especifica el tipo de vía.
- **Latitud**: de la ubicación de toma de datos.
- **Longitud**: de la ubicación de toma de datos.

	ï..FECHA	HORA	IDENTIFICADOR	PEATONES	NUMERO_DISTRITO	DISTRITO
1	01/01/2020 0:00	0:00	PERM_PEA02_PM01	1269	1	Centro
2	01/01/2020 1:00	1:00	PERM_PEA02_PM01	1137	1	Centro
3	01/01/2020 2:00	2:00	PERM_PEA02_PM01	843	1	Centro
4	01/01/2020 3:00	3:00	PERM_PEA02_PM01	711	1	Centro
5	01/01/2020 4:00	4:00	PERM_PEA02_PM01	476	1	Centro
6	01/01/2020 5:00	5:00	PERM_PEA02_PM01	600	1	Centro
7	01/01/2020 6:00	6:00	PERM_PEA02_PM01	662	1	Centro
8	01/01/2020 7:00	7:00	PERM_PEA02_PM01	1211	1	Centro
9	01/01/2020 8:00	8:00	PERM_PEA02_PM01	1676	1	Centro
10	01/01/2020 9:00	9:00	PERM_PEA02_PM01	1491	1	Centro
11	01/01/2020 10:00	10:00	PERM_PEA02_PM01	505	1	Centro
12	01/01/2020 11:00	11:00	PERM_PEA02_PM01	3481	1	Centro
13	01/01/2020 12:00	12:00	PERM_PEA02_PM01	6339	1	Centro
14	01/01/2020 13:00	13:00	PERM_PEA02_PM01	6086	1	Centro
15	01/01/2020 14:00	14:00	PERM_PEA02_PM01	9340	1	Centro
16	01/01/2020 15:00	15:00	PERM_PEA02_PM01	6957	1	Centro
17	01/01/2020 16:00	16:00	PERM_PEA02_PM01	4954	1	Centro
18	01/01/2020 17:00	17:00	PERM_PEA02_PM01	5804	1	Centro
19	01/01/2020 18:00	18:00	PERM_PEA02_PM01	9153	1	Centro
20	01/01/2020 19:00	19:00	PERM_PEA02_PM01	10333	1	Centro
21	01/01/2020 20:00	20:00	PERM_PEA02_PM01	14241	1	Centro
22	01/01/2020 21:00	21:00	PERM_PEA02_PM01	9064	1	Centro
23	01/01/2020 22:00	22:00	PERM_PEA02_PM01	4269	1	Centro
24	01/01/2020 23:00	23:00	PERM_PEA02_PM01	2145	1	Centro
25	02/01/2020 0:00	0:00	PERM_PEA02_PM01	1269	1	Centro
26	02/01/2020 1:00	1:00	PERM_PEA02_PM01	1202	1	Centro

Showing 1 to 27 of 166,896 entries, 12 total columns

Figura 3.1: Conjunto de datos inicial primeras 6 columnas

NOMBRE_VIAL	NUMERO	CODIGO_POSTAL	OBSERVACIONES_DIRECCION	LATITUD	LONGITUD
Calle Fuencarral	22	28004	Calle peatonal	40.42201	-3.700892
Calle Fuencarral	22	28004	Calle peatonal	40.42201	-3.700892
Calle Fuencarral	22	28004	Calle peatonal	40.42201	-3.700892
Calle Fuencarral	22	28004	Calle peatonal	40.42201	-3.700892
Calle Fuencarral	22	28004	Calle peatonal	40.42201	-3.700892
Calle Fuencarral	22	28004	Calle peatonal	40.42201	-3.700892
Calle Fuencarral	22	28004	Calle peatonal	40.42201	-3.700892
Calle Fuencarral	22	28004	Calle peatonal	40.42201	-3.700892
Calle Fuencarral	22	28004	Calle peatonal	40.42201	-3.700892
Calle Fuencarral	22	28004	Calle peatonal	40.42201	-3.700892
Calle Fuencarral	22	28004	Calle peatonal	40.42201	-3.700892
Calle Fuencarral	22	28004	Calle peatonal	40.42201	-3.700892
Calle Fuencarral	22	28004	Calle peatonal	40.42201	-3.700892
Calle Fuencarral	22	28004	Calle peatonal	40.42201	-3.700892
Calle Fuencarral	22	28004	Calle peatonal	40.42201	-3.700892
Calle Fuencarral	22	28004	Calle peatonal	40.42201	-3.700892
Calle Fuencarral	22	28004	Calle peatonal	40.42201	-3.700892
Calle Fuencarral	22	28004	Calle peatonal	40.42201	-3.700892
Calle Fuencarral	22	28004	Calle peatonal	40.42201	-3.700892
Calle Fuencarral	22	28004	Calle peatonal	40.42201	-3.700892
Calle Fuencarral	22	28004	Calle peatonal	40.42201	-3.700892
Calle Fuencarral	22	28004	Calle peatonal	40.42201	-3.700892
Calle Fuencarral	22	28004	Calle peatonal	40.42201	-3.700892
Calle Fuencarral	22	28004	Calle peatonal	40.42201	-3.700892
Calle Fuencarral	22	28004	Calle peatonal	40.42201	-3.700892
Calle Fuencarral	22	28004	Calle peatonal	40.42201	-3.700892

Figura 3.2: Conjunto de datos inicial siguientes 6 columnas

3.1. Tratamiento de datos

Carga de datos y selección de variables

Procedemos a cargar los datos del csv en R dentro de un dataframe sin realizar ninguna modificación sobre sus datos, a continuación, seleccionamos las columnas que vamos a utilizar en nuestro desarrollo y que consideramos relevantes, la aplicación de nuestro algoritmo se pretende hacer sobre el número de peatones medio en el mes de agosto, a las veinte horas en las calles del centro de Madrid, por tanto las variables que se recogen en este primer tratamiento son nombre vial, peatones, día, mes, año y hora. Podemos observar el resultado de este proceso y el la tabla generada en la figura 3.3.

	NOMBRE_VIAL	PEATONES	ï..FECHA	HORA	dia	mes	anio
1	Calle Fuencarral	1269	01/01/2020 0:00	0:00	1	1	2020
2	Calle Fuencarral	1137	01/01/2020 1:00	1:00	1	1	2020
3	Calle Fuencarral	843	01/01/2020 2:00	2:00	1	1	2020
4	Calle Fuencarral	711	01/01/2020 3:00	3:00	1	1	2020
5	Calle Fuencarral	476	01/01/2020 4:00	4:00	1	1	2020
6	Calle Fuencarral	600	01/01/2020 5:00	5:00	1	1	2020
7	Calle Fuencarral	662	01/01/2020 6:00	6:00	1	1	2020
8	Calle Fuencarral	1211	01/01/2020 7:00	7:00	1	1	2020
9	Calle Fuencarral	1676	01/01/2020 8:00	8:00	1	1	2020
10	Calle Fuencarral	1491	01/01/2020 9:00	9:00	1	1	2020
11	Calle Fuencarral	505	01/01/2020 10:00	10:00	1	1	2020
12	Calle Fuencarral	3481	01/01/2020 11:00	11:00	1	1	2020
13	Calle Fuencarral	6339	01/01/2020 12:00	12:00	1	1	2020
14	Calle Fuencarral	6086	01/01/2020 13:00	13:00	1	1	2020
15	Calle Fuencarral	9340	01/01/2020 14:00	14:00	1	1	2020
16	Calle Fuencarral	6957	01/01/2020 15:00	15:00	1	1	2020
17	Calle Fuencarral	4954	01/01/2020 16:00	16:00	1	1	2020
18	Calle Fuencarral	5804	01/01/2020 17:00	17:00	1	1	2020
19	Calle Fuencarral	9153	01/01/2020 18:00	18:00	1	1	2020
20	Calle Fuencarral	10333	01/01/2020 19:00	19:00	1	1	2020
21	Calle Fuencarral	14241	01/01/2020 20:00	20:00	1	1	2020
22	Calle Fuencarral	9064	01/01/2020 21:00	21:00	1	1	2020

Figura 3.3: Selección de columnas

Filtrado y cálculo de media

Aplicamos un primer tratamiento sobre la información recogida, filtramos el conjunto de datos con los siguientes criterios anteriormente explicados, tomamos todos los registros que tienen como mes agosto y como hora de medida las 20 horas, obteniendo la siguiente Figura 3.4.

	NOMBRE_VIAL	PEATONES	ï..FECHA	HORA	dia	mes	anio
1	Calle Fuencarral	4570	01/08/2020 20:00	20:00	1	8	2020
2	Calle Fuencarral	3419	02/08/2020 20:00	20:00	2	8	2020
3	Calle Fuencarral	2973	03/08/2020 20:00	20:00	3	8	2020
4	Calle Fuencarral	4356	04/08/2020 20:00	20:00	4	8	2020
5	Calle Fuencarral	4001	05/08/2020 20:00	20:00	5	8	2020
6	Calle Fuencarral	3993	06/08/2020 20:00	20:00	6	8	2020
7	Calle Fuencarral	2992	07/08/2020 20:00	20:00	7	8	2020
8	Calle Fuencarral	0	08/08/2020 20:00	20:00	8	8	2020
9	Calle Fuencarral	4595	09/08/2020 20:00	20:00	9	8	2020
10	Calle Fuencarral	4646	10/08/2020 20:00	20:00	10	8	2020
11	Calle Fuencarral	1472	11/08/2020 20:00	20:00	11	8	2020
12	Calle Fuencarral	4749	12/08/2020 20:00	20:00	12	8	2020
13	Calle Fuencarral	4111	13/08/2020 20:00	20:00	13	8	2020
14	Calle Fuencarral	2459	14/08/2020 20:00	20:00	14	8	2020
15	Calle Fuencarral	5005	15/08/2020 20:00	20:00	15	8	2020
16	Calle Fuencarral	3690	16/08/2020 20:00	20:00	16	8	2020
17	Calle Fuencarral	4024	17/08/2020 20:00	20:00	17	8	2020
18	Calle Fuencarral	4011	18/08/2020 20:00	20:00	18	8	2020
19	Calle Fuencarral	3823	19/08/2020 20:00	20:00	19	8	2020
20	Calle Fuencarral	3564	20/08/2020 20:00	20:00	20	8	2020
21	Calle Fuencarral	3638	21/08/2020 20:00	20:00	21	8	2020
22	Calle Fuencarral	4164	22/08/2020 20:00	20:00	22	8	2020

Showing 1 to 23 of 589 entries, 7 total columns

Figura 3.4: Filtrado por mes y hora

Para el cálculo de la media se ha agrupado por calle y por día mientras que la función se aplicó sobre la columna de peatones. Figura 3.5. A continuación, utilizamos la función reshape para remodelar nuestro dataframe obteniendo como nuevas columnas cada uno de los días del mes desde el 1 hasta el 31 de agosto para cada una de las calles, para posteriormente sumar las

columnas pertenecientes a cada día de la semana, obteniendo únicamente ocho columnas, nombre de la calle, y el día correspondiente de la semana. Figura 3.6

	Group.1	Group.2	x
1	Alberto Aguilera	1	114.0
2	Calle Alcala	1	264.0
3	Calle Atocha	1	63.0
4	Calle Bailen	1	138.0
5	Calle Fuencarral	1	4570.0
6	Calle Genova	1	97.0
7	Calle Hortaleza	1	198.0
8	Calle Huertas	1	30.0
9	Calle Mayor	1	108.0
10	Calle Princesa	1	266.0
11	Calle San Bernardo	1	136.0
12	Calle Toledo	1	1424.0
13	Carrera de San Jeronimo	1	272.0
14	Gran Via	1	1031.0
15	Madrid Rio. Puente de Segovia con Paseo Ermita del Santo ...	1	68.0
16	Paseo de Recoletos	1	129.0
17	Plaza del Emperador Carlos V	1	0.0
18	Ronda de Valencia	1	205.0
19	Alberto Aguilera	2	173.0
20	Calle Alcala	2	116.0
21	Calle Atocha	2	66.0
22	Calle Bailen	2	171.0

Showing 1 to 23 of 558 entries, 3 total columns

Figura 3.5: Cálculo de media

	Group.1	x.1	x.2	x.3	x.4	x.5	x.6	x.7	x.8	x.9	x.10	x.11	x.12	x.13	x.14	x.15	x.16	x.17	x.18	x.19	x.20	x.21	x.22
1	Alberto Aguilera	114	173	264.0	258.0	266.0	260	169.0	18	155.0	565	574	242	291	222	133	215.0	272	187	151.0	129.0	85	14
2	Calle Alcala	264	116	132.0	133.0	82.0	67	93.0	266	97.0	65	50	106	129	28	256	109.0	148	108	107.0	178.0	151	179
3	Calle Atocha	63	66	157.0	132.0	116.0	142	134.0	128	17.0	56	123	242	275	0	123	168.0	274	205	253.0	305.0	201	182
4	Calle Bailen	138	171	143.0	142.0	48.0	89	111.0	184	121.0	108	50	160	211	146	292	230.0	201	143	94.0	194.0	142	155
5	Calle Fuencarral	4570	3419	2973.0	4356.0	4001.0	3993	2992.0	0	4595.0	4646	1472	4749	4111	2459	5005	3690.0	4024	4011	3823.0	3564.0	3638	4164
6	Calle Genova	97	65	64.0	86.0	94.0	74	154.0	173	40.0	209	149	143	137	134	176	37.0	166	172	163.0	89.0	135	183
7	Calle Hortaleza	198	28	20.0	15.0	82.0	102	71.0	31	39.0	113	18	228	269	147	180	183.0	162	132	215.0	152.0	122	218
8	Calle Huertas	30	34	46.0	39.0	23.0	37	14.0	42	20.0	39	37	56	21	37	47	37.0	15	58	25.0	27.0	33	48
9	Calle Mayor	108	153	118.0	106.0	95.0	119	131.0	185	151.0	100	139	121	136	174	119	141.0	107	117	111.0	82.0	63	192
10	Calle Princesa	266	209	288.0	348.0	418.0	208	122.0	307	238.0	173	17	102	37	227	414	119.0	175	199	331.0	250.0	54	96
11	Calle San Bernardo	136	84	122.0	118.0	98.0	117	167.0	106	67.0	121	0	144	135	156	127	80.0	44	104	96.0	173.0	79	81
12	Calle Toledo	1424	1006	920.0	882.0	897.0	859	851.0	1368	1051.0	890	943	989	897	882	1424	1122.0	920	982	966.0	874.0	913	1244
13	Carrera de San Jeronimo	272	150	215.0	109.0	158.0	60	59.0	134	206.0	118	110	247	193	294	337	183.0	183	244	171.0	398.0	377	474
14	Gran Via	1031	1267	1308.5	1213.5	1154.5	989	939.5	1793	1083.5	1192	603	1350	1123	1187	1416	1088.5	1275	1302	977.5	1019.5	986	1203
15	Madrid Rio. Puente de Segovia con Paseo Ermita del Santo ...	68	140	181.0	138.0	136.0	123	88.0	105	105.0	206	34	234	198	153	141	272.0	279	252	176.0	202.0	207	181
16	Paseo de Recoletos	129	81	83.0	84.0	94.0	88	91.0	119	74.0	83	86	82	82	141	111	80.0	82	107	65.0	47.0	67	85
17	Plaza del Emperador Carlos V	0	0	332.0	0.0	318.0	370	324.0	278	388.0	83	370	370	344	329	285	348.0	358	0	344.0	358.0	267	452
18	Ronda de Valencia	205	240	137.0	382.0	357.0	205	278.0	326	274.0	341	335	367	332	166	276	256.0	231	327	185.0	289.0	238	277

Figura 3.6: Reshape dataframe

Dataframe final

Una vez realizado todo el tratamiento de datos sobre nuestro conjunto inicial, nos encontramos con la información completamente limpia y lista para ejecutar los algoritmos K-Means y Hierarchical clustering de manera individual.

	calle	lunes	martes	miercoles	jueves	viernes	sabado	domingo
1	Alberto Aguilera	2346	1181	820.0	819.0	655	513	990
2	Calle Alcala	692	396	380.0	459.0	434	1209	631
3	Calle Atocha	1079	596	889.0	999.0	558	851	721
4	Calle Bailen	924	486	419.0	621.0	509	940	1189
5	Calle Fuencarral	19166	13680	15972.0	14998.0	13080	20062	19912
6	Calle Genova	771	518	507.0	401.0	537	843	181
7	Calle Hortaleza	582	321	632.0	640.0	595	909	492
8	Calle Huertas	198	174	133.0	101.0	104	212	163
9	Calle Mayor	525	455	446.0	390.0	456	831	626
10	Calle Princesa	1241	607	1069.0	742.0	708	1573	1040
11	Calle San Bernardo	598	330	370.0	565.0	543	620	428
12	Calle Toledo	4641	3750	3780.0	3235.0	3413	6581	5208
13	Carrera de San Jeronimo	848	680	610.0	901.0	1091	1399	984
14	Gran Via	5978	4171	4200.5	4090.5	4250	7621	5925
15	Madrid Rio. Puente de Segovia con Paseo Ermita del Santo ...	1322	671	810.0	716.0	674	822	1136
16	Paseo de Recoletos	455	316	361.0	280.0	419	559	370
17	Plaza del Emperador Carlos V	1557	775	1482.0	1274.0	1148	1313	1660
18	Ronda de Valencia	1373	1194	1207.0	1100.0	1014	1397	1317

Showing 1 to 18 of 18 entries, 8 total columns

Figura 3.7: Dataframe final

4. Desarrollo del proyecto

4.1. Ejecución K-means

A continuación se muestran una serie de iteraciones que se probó sobre el algoritmo kmeans antes de generar la clasificación final usando el numero de clusters óptimos.

1º Iteración

En la primera iteración del algoritmo de K-Means se fijo como número de clusters a generar en 4, cómo podemos observar en la Figura 4.1 el algoritmo ha realizado la clasificación sobre nuestro conjunto de datos dando lugar a cuatro agrupaciones, en los clusters uno y dos apreciamos un gran número de individuos mientras que los conjuntos tres y cuatro tienen uno y dos individuos respectivamente.

```
K-means clustering with 4 clusters of sizes 8, 7, 1, 2

Cluster means:
      lunes     martes  miercoles     jueves    viernes    sabado   domingo
1   593.125   374.5000   406.0000   432.1250   449.6250   765.375   510.000
2  1395.143   814.8571   983.8571   935.8571   835.4286  1124.000  1121.143
3 19166.000 13680.0000 15972.0000 14998.0000 13080.0000 20062.000 19912.000
4  5309.500  3960.5000  3990.2500  3662.7500  3831.5000  7101.000  5566.500

Clustering vector:
 [1] 2 1 2 1 3 1 1 1 1 2 1 4 2 4 2 1 2 2

Within cluster sum of squares by cluster:
[1] 2328016 4330618       0 2584884
 (between_SS / total_SS =  99.5 %)
```

Figura 4.1: Resultado cuatro clusters

De modo que no podemos considerar una clasificación relevante la creación de 4 clusters.

2º Iteración

Nuevamente fijamos el número de clusters a generar por el algoritmo K-Means, en esta ocasión probamos la clasificación en 5 grupos.

```
K-means clustering with 5 clusters of sizes 5, 3, 2, 7, 1

Cluster means:
       lunes     martes  miercoles     jueves    viernes     sabado   domingo
1  1082.8000   608.0000   759.4000   795.8000   708.0000  1117.0000  1014.000
2  1758.6667  1050.0000  1169.6667  1064.3333   939.0000  1074.3333  1322.333
3  5309.5000  3960.5000  3990.2500  3662.7500  3831.5000  7101.0000  5566.500
4   545.8571   358.5714   404.1429   405.1429   441.1429   740.4286   413.000
5 19166.0000 13680.0000 15972.0000 14998.0000 13080.0000 20062.0000 19912.000

Clustering vector:
 [1] 2 4 1 1 5 4 4 4 4 1 4 3 1 3 1 4 2 2

Within cluster sum of squares by cluster:
[1] 1351815 1805149 2584884 1581943       0
 (between_SS / total_SS =  99.6 %)
```

Figura 4.2: Resultado cinco clusters

A pesar de que se aprecia una mayor distribución de los elementos dentro de los clusters podemos encontrar dos conjuntos que tienen la misma problemática que en el caso anterior, el cluster cinco contiene únicamente un individuo mientras que el cluster cuatro se encuentra sobre-poblado.

Elección de clusters óptimos

En nuestra iteración final se decidió ejecutar nuestro algoritmo varias veces y tomar la precisión generada para decidir nuestro número de cluster óptimo, de esta manera se toma aquel valor que se haya generado como primer máximo durante la ejecución del algoritmo sobre nuestro conjunto de datos.

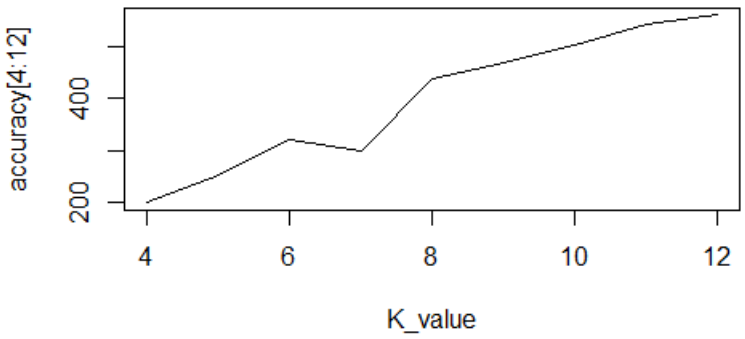

Figura 4.3: Cluster óptimo

Tras la ejecución con nuestro primer máximo como número óptimo de cluster se consiguieron los siguientes resultados.

	posicion	observacion	cluster_hoja
1	1	6	4
2	2	16	3
3	3	4	5
4	4	9	5
5	5	11	2
6	6	3	5
7	7	15	1
8	8	2	1
9	9	13	5
10	10	7	4
11	11	8	5
12	12	10	6
13	13	18	3
14	14	1	6
15	15	12	5
16	16	14	5
17	17	17	4
18	18	5	4

Showing 1 to 18 of 18 entries, 7 total column

Figura 4.4: Resultado seis clusters

Como podemos apreciar en la Figura 4.4, la distribución de los elementos es ligeramente mejor que los casos anteriores pero a pesar de esto, seguimos encontrando agrupaciones de un solo elemento por lo que se puede concluir que aunque la clasificación es buena, en el problema que intentamos resolver no arroja una solución relevante, por tanto, recurrimos al algoritmo jerárquico para intentar mejorar nuestras agrupaciones.

4.2. Ejecución Hierarchical clustering

Ejecutamos el algoritmo jerárquico sobre nuestro conjunto de datos, para ello y como primer paso convertimos nuestro dataframe en una matriz usando la función *as.matrix* a continuación, obtenemos la matriz de distancias de nuestro conjunto de datos y ejecutamos *hclust*.

Obtención de ruta

Una vez aplicado el algoritmo a nuestro conjunto de datos se procede a obtener la ruta desde la raíz hasta las hojas, consiguiendo de esta manera el camino completo de cada observación, la salida generada contiene los valores 1 y 2 que representan las direcciones izquierda y derecha dentro del dendograma.

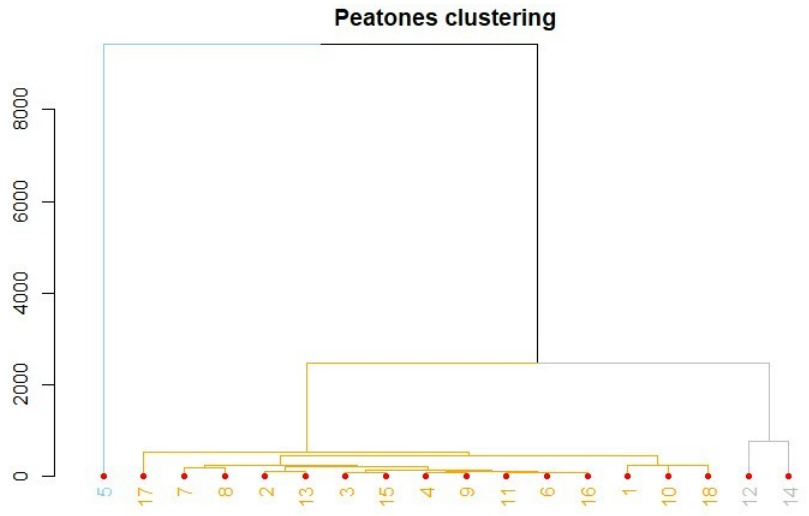

Figura 4.5: Dendograma generado sin etiquetas

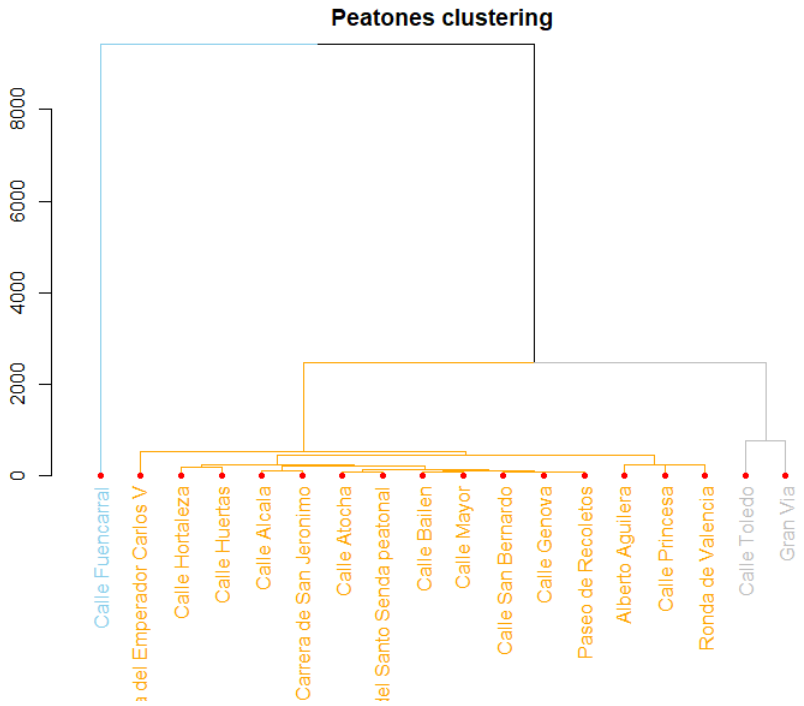

Figura 4.6: Dendograma generado con etiquetas

Procedemos a ordenar el conjunto de datos en función del tamaño de su ruta, de esta manera, tenemos aquellas observaciones con una mayor profundidad en el árbol como primeros elementos de nuestra tabla.

	posicion	observacion	ruta
1	1	6	2, 1, 2, 1, 2, 2, 2, 2, 2, 1
2	2	16	2, 1, 2, 1, 2, 2, 2, 2, 2, 2
3	3	4	2, 1, 2, 1, 2, 2, 2, 1, 1
4	4	9	2, 1, 2, 1, 2, 2, 2, 1, 2
5	5	11	2, 1, 2, 1, 2, 2, 2, 2, 1
6	6	3	2, 1, 2, 1, 2, 2, 1, 1
7	7	15	2, 1, 2, 1, 2, 2, 1, 2
8	8	2	2, 1, 2, 1, 2, 1, 1
9	9	13	2, 1, 2, 1, 2, 1, 2
10	10	7	2, 1, 2, 1, 1, 1
11	11	8	2, 1, 2, 1, 1, 2
12	12	10	2, 1, 2, 2, 2, 1
13	13	18	2, 1, 2, 2, 2, 2
14	14	1	2, 1, 2, 2, 1
15	15	12	2, 2, 1
16	16	14	2, 2, 2
17	17	17	2, 1, 1
18	18	5	1

Figura 4.7: Salida dendograma

De esta manera ya tenemos los datos suficientes para generar un nuevo dataframe sobre el que trabajar, que contiene las columnas posición, observación, ruta, tamaño, cluster hoja, cluster nuevo, agrupado.

	posicion	observacion	ruta	tamanio	cluster_hoja	cluster_nuevo	agrupado
1	1	6	2, 1, 2, 1, 2, 2, 2, 2, 2, 1	10	4	0	FALSE
2	2	16	2, 1, 2, 1, 2, 2, 2, 2, 2, 2	10	3	0	FALSE
3	3	4	2, 1, 2, 1, 2, 2, 2, 1, 1	9	5	0	FALSE
4	4	9	2, 1, 2, 1, 2, 2, 2, 1, 2	9	5	0	FALSE
5	5	11	2, 1, 2, 1, 2, 2, 2, 2, 1	9	2	0	FALSE
6	6	3	2, 1, 2, 1, 2, 2, 1, 1	8	5	0	FALSE
7	7	15	2, 1, 2, 1, 2, 2, 1, 2	8	1	0	FALSE
8	8	2	2, 1, 2, 1, 2, 1, 1	7	1	0	FALSE
9	9	13	2, 1, 2, 1, 2, 1, 2	7	5	0	FALSE
10	10	7	2, 1, 2, 1, 1, 1	6	4	0	FALSE
11	11	8	2, 1, 2, 1, 1, 2	6	5	0	FALSE
12	12	10	2, 1, 2, 2, 2, 1	6	6	0	FALSE
13	13	18	2, 1, 2, 2, 2, 2	6	3	0	FALSE
14	14	1	2, 1, 2, 2, 1	5	6	0	FALSE
15	15	12	2, 2, 1	3	5	0	FALSE
16	16	14	2, 2, 2	3	5	0	FALSE
17	17	17	2, 1, 1	3	4	0	FALSE
18	18	5	1	1	4	0	FALSE

Showing 1 to 18 of 18 entries, 7 total columns

Figura 4.8: Nuevo dataframe sin agrupar

4.3. Algoritmo Merge

Partiendo de la definición de nomenclatura e implementación explicada en el capitulo 3, procedemos a la implementación del nuevo algoritmo Merge.

En primer lugar como se observa en la Figura 4.9 comprobamos aquellas observaciones del mismo tamaño k y que comparten prefijo que se extrajo de la ruta generada por el algoritmo Hierarchical Clustering, en este caso las observaciones O_6 y O_{16} comparten una misma raíz lo que consideramos ser observaciones hermanas y por ende se agrupan en un cluster y se procede a buscar aquellas observaciones que se consideran su primos, en este caso utilizamos la columna cluster hoja que se obtuvo de la clasificación realizada por K-Means, todos aquellos registros que compartan los mismos cluster de O_6 y O_{16} también se agrupan en el nuevo cluster y se pone a TRUE la columna agrupado.

posicion	observacion	ruta	tamanio	cluster_hoja	cluster_nuevo	agrupado
1	6	2, 1, 2, 1, 2, 2, 2, 2, 2	1	10	4	0 FALSE
2	16	2, 1, 2, 1, 2, 2, 2, 2, 2	2	10	3	0 FALSE
3	4	2, 1, 2, 1, 2, 2, 2, 1, 1		9	5	0 FALSE
4	9	2, 1, 2, 1, 2, 2, 2, 1, 2		9	5	0 FALSE
5	11	2, 1, 2, 1, 2, 2, 2, 2, 1		9	2	0 FALSE
6	3	2, 1, 2, 1, 2, 2, 1, 1		8	5	0 FALSE
7	15	2, 1, 2, 1, 2, 2, 1, 2		8	1	0 FALSE
8	2	2, 1, 2, 1, 2, 1, 1		7	1	0 FALSE
9	13	2, 1, 2, 1, 2, 1, 2		7	5	0 FALSE
10	7	2, 1, 2, 1, 1, 1		6	4	0 FALSE
11	8	2, 1, 2, 1, 1, 2		6	5	0 FALSE
12	10	2, 1, 2, 2, 2, 1		6	6	0 FALSE
13	18	2, 1, 2, 2, 2, 2		6	3	0 FALSE
14	1	2, 1, 2, 2, 1		5	6	0 FALSE
15	12	2, 2, 1		3	5	0 FALSE
16	14	2, 2, 2		3	5	0 FALSE
17	17	2, 1, 1		3	4	0 FALSE
18	5	1		1	4	0 FALSE

posicion	observacion	ruta	tamanio	cluster_hoja	cluster_nuevo	agrupado
1	6	2, 1, 2, 1, 2, 2, 2, 2, 2	1	10	4	0 TRUE
2	16	2, 1, 2, 1, 2, 2, 2, 2, 2	2	10	3	0 TRUE
3	4	2, 1, 2, 1, 2, 2, 2, 1, 1		9	5	0 FALSE
4	9	2, 1, 2, 1, 2, 2, 2, 1, 2		9	5	0 FALSE
5	11	2, 1, 2, 1, 2, 2, 2, 2, 1		9	2	0 FALSE
6	3	2, 1, 2, 1, 2, 2, 1, 1		8	5	0 FALSE
7	15	2, 1, 2, 1, 2, 2, 1, 2		8	1	0 FALSE
8	2	2, 1, 2, 1, 2, 1, 1		7	1	0 FALSE
9	13	2, 1, 2, 1, 2, 1, 2		7	5	0 FALSE
10	7	2, 1, 2, 1, 1, 1		6	4	0 TRUE
11	8	2, 1, 2, 1, 1, 2		6	5	0 FALSE
12	10	2, 1, 2, 2, 2, 1		6	6	0 FALSE
13	18	2, 1, 2, 2, 2, 2		6	3	0 TRUE
14	1	2, 1, 2, 2, 1		5	6	0 FALSE
15	12	2, 2, 1		3	5	0 FALSE
16	14	2, 2, 2		3	5	0 FALSE
17	17	2, 1, 1		3	4	0 TRUE
18	5	1		1	4	0 TRUE

Figura 4.9: Búsqueda de hermanos

A continuación, se procede a comprobar si las observaciones de un tamaño k pueden ser o no hermanos con algún elemento de $k-1$ y todavía no han sido agrupados, al igual que en el primer paso, se comprueba que poseen el mismo prefijo, en la Figura 4.10 se pude ver que las observaciones O_6, O_{16} y O_{11} cumplen dicha condición y por tanto buscamos sus primos para el $cluster = 2$.

posicion	observacion	ruta	tamanio	cluster_hoja	cluster_nuevo	agrupado
1	6	2, 1, 2, 1, 2, 2, 2, 2, 2, 1	10	4	0	TRUE
2	16	2, 1, 2, 1, 2, 2, 2, 2, 2, 2	10	3	0	TRUE
3	4	2, 1, 2, 1, 2, 2, 2, 1, 1	9	5	0	FALSE
4	9	2, 1, 2, 1, 2, 2, 2, 1, 2	9	5	0	FALSE
5	11	2, 1, 2, 1, 2, 2, 2, 2, 1	9	2	0	FALSE
6	3	2, 1, 2, 1, 2, 2, 1, 1	8	5	0	FALSE
7	15	2, 1, 2, 1, 2, 2, 1, 2	8	1	0	FALSE
8	2	2, 1, 2, 1, 2, 1, 1	7	1	0	FALSE
9	13	2, 1, 2, 1, 2, 1, 2	7	5	0	FALSE
10	7	2, 1, 2, 1, 1, 1	6	4	0	TRUE
11	8	2, 1, 2, 1, 1, 2	6	5	0	FALSE
12	10	2, 1, 2, 2, 2, 1	6	6	0	FALSE
13	18	2, 1, 2, 2, 2, 2	6	3	0	TRUE
14	1	2, 1, 2, 2, 1	5	6	0	FALSE
15	12	2, 2, 1	3	5	0	FALSE
16	14	2, 2, 2	3	5	0	FALSE
17	17	2, 1, 1	3	4	0	TRUE
18	5	1	1	4	0	TRUE

posicion	observacion	ruta	tamanio	cluster_hoja	cluster_nuevo	agrupado
1	6	2, 1, 2, 1, 2, 2, 2, 2, 1	10	4	0	TRUE
2	16	2, 1, 2, 1, 2, 2, 2, 2, 2	10	3	0	TRUE
3	4	2, 1, 2, 1, 2, 2, 2, 1, 1	9	5	0	FALSE
4	9	2, 1, 2, 1, 2, 2, 2, 1, 2	9	5	0	FALSE
5	11	2, 1, 2, 1, 2, 2, 2, 2, 1	9	2	0	TRUE
6	3	2, 1, 2, 1, 2, 2, 1, 1	8	5	0	FALSE
7	15	2, 1, 2, 1, 2, 2, 1, 2	8	1	0	FALSE
8	2	2, 1, 2, 1, 2, 1, 1	7	1	0	FALSE
9	13	2, 1, 2, 1, 2, 1, 2	7	5	0	FALSE
10	7	2, 1, 2, 1, 1, 1	6	4	0	TRUE
11	8	2, 1, 2, 1, 1, 2	6	5	0	FALSE
12	10	2, 1, 2, 2, 2, 1	6	6	0	FALSE
13	18	2, 1, 2, 2, 2, 2	6	3	0	TRUE
14	1	2, 1, 2, 2, 1	5	6	0	FALSE
15	12	2, 2, 1	3	5	0	FALSE
16	14	2, 2, 2	3	5	0	FALSE
17	17	2, 1, 1	3	4	0	TRUE
18	5	1	1	4	0	TRUE

Figura 4.10: Búsqueda de primos

El siguiente paso consiste aplicar este proceso decrementando el valor de k, en la Figura 4.11 podemos observar como se va agrupando en cada nivel de tamaños de ruta en este caso agrupa aquellos elementos de k y los elementos que en *k-1* tienen el mismo prefijo y además también agrupa aquellos que comparten el mismo cluster generado en K-Means, llegando a un cluster compuesto por 15 observaciones que componen un nuevo conjunto generado por *Merge*, nuevamente se aplica este proceso hasta que no se pueda agrupar más.

posicion	observacion	ruta	tamanio	cluster_hoja	cluster_nuevo	agrupado
1	6	2, 1, 2, 1, 2, 2, 2, 2, 2, 1	10	4	0	TRUE
2	16	2, 1, 2, 1, 2, 2, 2, 2, 2, 2	10	3	0	TRUE
3	4	2, 1, 2, 1, 2, 2, 2, 1	1	9	5	0	FALSE
4	9	2, 1, 2, 1, 2, 2, 2, 1	2	9	5	0	FALSE
5	11	2, 1, 2, 1, 2, 2, 2, 2, 1	9	2	0	TRUE
6	3	2, 1, 2, 1, 2, 2, 1, 1	8	5	0	FALSE
7	15	2, 1, 2, 1, 2, 2, 1, 2	8	1	0	FALSE
8	2	2, 1, 2, 1, 2, 1, 1	7	1	0	FALSE
9	13	2, 1, 2, 1, 2, 1, 2	7	5	0	FALSE
10	7	2, 1, 2, 1, 1, 1	6	4	0	TRUE
11	8	2, 1, 2, 1, 1, 2	6	5	0	FALSE
12	10	2, 1, 2, 2, 2, 1	6	6	0	FALSE
13	18	2, 1, 2, 2, 2, 2	6	3	0	TRUE
14	1	2, 1, 2, 2, 1	5	6	0	FALSE
15	12	2, 2, 1	3	5	0	FALSE
16	14	2, 2, 2	3	5	0	FALSE
17	17	2, 1, 1	3	4	0	TRUE
18	5	1	1	4	0	TRUE

posicion	observacion	ruta	tamanio	cluster_hoja	cluster_nuevo	agrupado
1	6	2, 1, 2, 1, 2, 2, 2, 2, 2, 1	10	4	0	TRUE
2	16	2, 1, 2, 1, 2, 2, 2, 2, 2, 2	10	3	0	TRUE
3	4	2, 1, 2, 1, 2, 2, 2, 1, 1	9	5	0	TRUE
4	9	2, 1, 2, 1, 2, 2, 2, 1, 2	9	5	0	TRUE
5	11	2, 1, 2, 1, 2, 2, 2, 2, 1	9	2	0	TRUE
6	3	2, 1, 2, 1, 2, 2, 1, 1	8	5	0	TRUE
7	15	2, 1, 2, 1, 2, 2, 1, 2	8	1	0	FALSE
8	2	2, 1, 2, 1, 2, 1, 1	7	1	0	FALSE
9	13	2, 1, 2, 1, 2, 1, 2	7	5	0	TRUE
10	7	2, 1, 2, 1, 1, 1	6	4	0	TRUE
11	8	2, 1, 2, 1, 1, 2	6	5	0	TRUE
12	10	2, 1, 2, 2, 2, 1	6	6	0	FALSE
13	18	2, 1, 2, 2, 2, 2	6	3	0	TRUE
14	1	2, 1, 2, 2, 1	5	6	0	FALSE
15	12	2, 2, 1	3	5	0	TRUE
16	14	2, 2, 2	3	5	0	TRUE
17	17	2, 1, 1	3	4	0	TRUE
18	5	1	1	4	0	TRUE

Figura 4.11: Búsqueda de elementos hermanos y primos en $k-1$

5. Resultados de los algoritmos

Los resultados arrojados de manera individual por K-Means y Hierarchical clustering nos ha permitido observar la generación de lo que podríamos llamar dos tipos de agrupaciones, en el caso de K-Means agrupaciones numerosas pero de escasos individuos, mientras que en el caso de Hierarchical clustering pocas agrupaciones de muchos elementos.

En las figuras 5.1 y 5.2 podemos ver la clasificación final generada por Merge, en la columna cluster nuevo se aprecia la clasificación generada por *Merge* mediante la búsqueda de observaciones hermanas y primas, en este caso se ha generado 2 clusters de tamaño 14 y 4 respectivamente.

posicion	observacion	ruta	tamanio	cluster_hoja	cluster_nuevo	agrupado
1	6	2, 1, 2, 1, 2, 2, 2, 2, 1	10	4	4	TRUE
2	16	2, 1, 2, 1, 2, 2, 2, 2, 2	10	3	4	TRUE
3	4	2, 1, 2, 1, 2, 2, 2, 1, 1	9	5	4	TRUE
4	9	2, 1, 2, 1, 2, 2, 2, 1, 2	9	5	4	TRUE
5	11	2, 1, 2, 1, 2, 2, 2, 2, 1	9	2	4	TRUE
6	3	2, 1, 2, 1, 2, 2, 1, 1	8	5	4	TRUE
7	15	2, 1, 2, 1, 2, 2, 1, 2	8	1	3	TRUE
8	2	2, 1, 2, 1, 2, 1, 1	7	1	3	TRUE
9	13	2, 1, 2, 1, 2, 1, 2	7	5	4	TRUE
10	7	2, 1, 2, 1, 1, 1	6	4	4	TRUE
11	8	2, 1, 2, 1, 1, 2	6	5	4	TRUE
12	10	2, 1, 2, 2, 2, 1	6	6	3	FALSE
13	18	2, 1, 2, 2, 2, 2	6	3	4	TRUE
14	1	2, 1, 2, 2, 1	5	6	3	FALSE
15	12	2, 2, 1	3	5	4	TRUE
16	14	2, 2, 2	3	5	4	TRUE
17	17	2, 1, 1	3	4	4	TRUE
18	5	1	1	4	4	TRUE

Figura 5.1: Agrupación generada por Merge

posicion	observacion	ruta	tamanio	cluster_hoja	cluster_nuevo	agrupado
1	6	2, 1, 2, 1, 2, 2, 2, 2, 2, 1	10	4	5	TRUE
2	16	2, 1, 2, 1, 2, 2, 2, 2, 2, 2	10	3	5	TRUE
3	4	2, 1, 2, 1, 2, 2, 2, 1, 1	9	5	5	TRUE
4	9	2, 1, 2, 1, 2, 2, 2, 1, 2	9	5	5	TRUE
5	11	2, 1, 2, 1, 2, 2, 2, 2, 1	9	2	5	TRUE
6	3	2, 1, 2, 1, 2, 2, 1, 1	8	5	5	TRUE
7	15	2, 1, 2, 1, 2, 2, 1, 2	8	1	4	TRUE
8	2	2, 1, 2, 1, 2, 1, 1	7	1	4	TRUE
9	13	2, 1, 2, 1, 2, 1, 2	7	5	5	TRUE
10	7	2, 1, 2, 1, 1, 1	6	4	5	TRUE
11	8	2, 1, 2, 1, 1, 2	6	5	5	TRUE
12	10	2, 1, 2, 2, 2, 1	6	6	4	TRUE
13	18	2, 1, 2, 2, 2, 2	6	3	5	TRUE
14	1	2, 1, 2, 2, 1	5	6	4	TRUE
15	12	2, 2, 1	3	5	5	TRUE
16	14	2, 2, 2	3	5	5	TRUE
17	17	2, 1, 1	3	4	5	TRUE
18	5	1	1	4	5	TRUE

Figura 5.2: Agrupación Merge

El último paso que realiza nuestro algoritmo es comprobar el tamaño de los nuevos conjuntos generados, en aquellos cluster que la cantidad de elementos sea mayor que el número establecido en el paso 1, se procede a subdividir dichos conjuntos para adecuarse a esa restricción de tamaño. Como se puede apreciar en la Figura 5.3.

posicion	observacion	ruta	tamanio	cluster_hoja	cluster_nuevo	agrupado
1	6	2, 1, 2, 1, 2, 2, 2, 2, 2, 1	10	4	5	TRUE
2	16	2, 1, 2, 1, 2, 2, 2, 2, 2, 2	10	3	5	TRUE
3	4	2, 1, 2, 1, 2, 2, 2, 1, 1	9	5	5	TRUE
4	9	2, 1, 2, 1, 2, 2, 2, 1, 2	9	5	5	TRUE
5	11	2, 1, 2, 1, 2, 2, 2, 2, 1	9	2	6	TRUE
6	3	2, 1, 2, 1, 2, 2, 1, 1	8	5	6	TRUE
7	15	2, 1, 2, 1, 2, 2, 1, 2	8	1	4	TRUE
8	2	2, 1, 2, 1, 2, 1, 1	7	1	4	TRUE
9	13	2, 1, 2, 1, 2, 1, 2	7	5	6	TRUE
10	7	2, 1, 2, 1, 1, 1	6	4	6	TRUE
11	8	2, 1, 2, 1, 1, 2	6	5	7	TRUE
12	10	2, 1, 2, 2, 2, 1	6	6	4	TRUE
13	18	2, 1, 2, 2, 2, 2	6	3	7	TRUE
14	1	2, 1, 2, 2, 1	5	6	4	TRUE
15	12	2, 2, 1	3	5	7	TRUE
16	14	2, 2, 2	3	5	7	TRUE
17	17	2, 1, 1	3	4	8	TRUE
18	5	1	1	4	8	TRUE

Figura 5.3: Resultado final

El algoritmo Merge ha generado un número aceptable de agrupaciones de un tamaño similar por lo que consideramos que la fusión de los dos algoritmos ha mejorado la clasificación.

5.1. Análisis de resultados

Analizando los resultados sobre cada ejecución de los algoritmos es que las calles agrupadas son las siguientes:

- **Primer cluster**
  Senda peatonal, Alcalá
- **Segundo cluster**
  San Bernardo
- **Tercero cluster**
  Paseo de recoletos, Ronda de Valencia
- **Cuarto cluster**
  Genova, Hortaleza, Emperador Carlos V, Fuencarral
- **Quinto cluster**
  Bailen, Mayor, Atocha, Carrera de San Jerónimo, Huertas, Toledo, Gran vía
- **Sexto cluster**
  Princesa, Alberto Aguilera

Podemos apreciar la existencia de clusters de un solo elemento que no resultan relevantes para nuestro proposito.

Mientras que en la ejecución de Hierarchical clustering podemos observar grupos con las siguientes calles:

- **Primer cluster**
  Fuencarral
- **Segundo cluster**
  Senda peatonal, Alcalá, San Bernardo, Paseo de recoletos, Ronda de Valencia, Genova, Hortaleza, Emperador Carlos V, Fuencarral, Bailen, Mayor, Atocha, Carrera de San Jerónimo, Huertas, Princesa, Alberto Aguilera

- **Tercero cluster**
  Toledo, Gran vía

Hierarchical clustering genera grupos tanto numerosos como escasos de elementos.

Mediante nuestro nuevo algoritmo conseguimos 5 clusters con los siguientes elementos:

- **Primer cluster**
  Genova, Paseo de recoletos, Bailen, Mayor
- **Segundo cluster**
  Atocha, San Bernardo, San Jerónimo, Hortaleza
- **Tercero cluster**
  Senda peatonal, Alcalá, Princesa, Alberto Aguilera
- **Cuarto cluster**
  Huertas, Ronda de Valencia, Toledo, Gran Vía
- **Quinto cluster**
  Emperador Carlos V, Fuencarral

Podemos concluir que mediante la unión de K-Means y Hierarchical clustering *Merge* ha creado agrupaciones de datos con tamaños similares, considero que puede ser un buen algoritmo y que su aplicación puede resultar relevante.

6. Conclusión

Este proyecto se presentó como la implementación del algoritmo Merge presentado en el artículo [4] mediante el lenguaje R, se siguió cada una de las especificaciones establecidas en dicho artículo y se estuvo en constante comunicación con las autoras del trabajo [4].

De este trabajo he profundizado en los fundamentos de dos algoritmos muy utilizados, como son K-Means y Hierarchical clustering, y he aprendido como se puede aprovechar la utilidad de estos algoritmos para crear un algoritmo nuevo que se enfoca en el tamaño de los clusters.

Todo este desarrollo me ha ayudado a entender que los algoritmos ya desarrollados, en general no se pueden siempre aplicar y de ello, la importancia de la búsqueda y desarrollo de nuevos métodos de resolución de problemas.

Una vez implementado el algoritmo y utilizado con un conjunto de datos de prueba podemos concluir que se ha alcanzado los objetivos propuestos en este proyecto y confío en que en el futuro, *Merge* sea de gran utilidad en la comunidad científica.

Bibliografía

[1] Bhanu yerra's blog - a place to collect odd items and ruminate over, yes you got it, odd items! https://mlbhanuyerra.github.io/2018-02-19-Clustering-K-means/, 2022.

[2] Knn and kmeans. https://harshkr21august.medium.com/knn-and-kmeans-b741dfccb69, 2019.

[3] Unsupervised learning-hierarchical clustering. https://mmsubra1.medium.com/unsupervised-learning-hierarchical-clustering-3dc3cfbe100, 2020.

[4] Victoria López and Sonia Estévez-Martín. A cluster merge algorithm to classify on/offline records of researchers in programming training. *Implementation of a new classification algorithm in the R language.*, 2020.

[5] Ross Ihaka and Robert Gentleman. R: a language for data analysis and graphics. *Journal of computational and graphical statistics*, 5(3):299–314, 1996.

[6] J Allaire. Rstudio: integrated development environment for r. *Boston, MA*, 770(394):165–171, 2012.

[7] Honorio Enrique Crespo Díaz Alejo. Portal de datos abiertos del ayuntamiento de madrid. *Consultor de los ayuntamientos y de los juzgados: Revista técnica especializada en administración local y justicia municipal*, (3):128–144, 2020.

[8] Fernando Caparrini. Algoritmos de clustering - fernando sancho caparrini. *Algorithm Clustering*, 2020.

[9] Jose Heras. Clustering (agrupamiento), k-means con ejemplos en python - iartificial.net. https://www.iartificial.net/clustering-agrupamiento-kmeans-ejemplos-en-python/#K-Means, 2022.

[10] K-means. https://www.unioviedo.es/compnum/laboratorios_py/kmeans/kmeans.html#kmeans.