UNIVERSIDAD
DE LA REPUBLICA
URUGUAY

Propuesta de algoritmos para aglomeración jerárquica de datos longitudinales y/o grandes volúmenes de datos, bajo condición de existencia de observaciones repetidas.

Cindy Abreo - Cecilia Paciel

Tutores: Juan José Goyeneche - Elena Vernazza

Licenciatura en Estadística
Facultad de Ciencias Económicas y de Administración
Universidad de la República

Montevideo – Uruguay
Noviembre de 2020

INTEGRANTES DEL TRIBUNAL DE DEFENSA DE

Juan José Goyeneche

Elena Vernazza

Laura Nalbarte

Leonardo Moreno

Montevideo – Uruguay
Noviembre de 2020

RESUMEN

El propósito del trabajo es presentar un procedimiento de segmentación aglomerativo jerárquico que en determinadas condiciones permite reducir sustancialmente la dimensión de las matrices de distancia que se utilizan para la implementación del mismo. Esto posibilita manipular y analizar un mayor volumen de datos, lo que generalmente es una limitación operativa que se presenta.

A su vez se trabaja con datos provenientes de trayectorias secuenciales, que se caracterizan por ser de índole cualitativos, presentando repeticiones de las observaciones.

Bajo estas condiciones de volumen y naturaleza de las observaciones, se evalúa y propone una técnica que permite el tratamiento de datos longitudinales tratados como datos categóricos, y una metodología de segmentación a partir de las características antes mencionadas.

Palabras claves:
Aglomerativo Jerárquico, Agnes, Cluster, Optimal Matching Analysis, Segmentación.

Lista de figuras

Lista de tablas

Tabla de contenidos

Capítulo 1

Introducción

En el presente trabajo se proponen dos técnicas, una de ellas para la generación de una matriz de disimilaridades a partir de datos longitudinales y principalmente de naturaleza cualitativa. La segunda técnica ofrece una solución de clusterización aglomerativa jerárquica disminuyendo costos computacionales, e inclusive logrando obtener una resolución del problema analizado, puesto que existen casos en que los algoritmos de segmentación no son capaces de brindar respuesta por no poder manejar computacionalmente determinada cantidad de datos.

La primer formulación se basa en la implementación del método **Optimal Matching Analysis (OMA)**, planteando un método adaptativo **Optimal Matching Analysis Reducido (OMAR)**, con el fin de lograr manipular datos reduciendo el costo computacional que implica utilizar la técnica original. Tanto **OMA** como **OMAR** tienen como principal objetivo alinear secuencias, obtener distancias o disimilaridades, para luego utilizar como insumo en procedimientos como por ejemplo, de segmentación.

Bajo estas circunstancias, en este trabajo se propone la generación de matrices de disimilaridades que cumplen con las limitaciones y características que permiten implementar la segunda etapa, la clusterización aglomerativa jerárquica **Agnes** [Kaufman, 2008]. La propuesta es un adaptativo **AGNES-R** a la técnica **Agnes**, basado en la formulación recursiva de *Lance y Williams* [Lance y Williams, 1966], donde se logra una representación reducida del problema disminuyendo los costos computacionales. El algoritmo **AGNES-R** pue-

de ser considerado como la técnica de segmentación mencionada que secunda al algoritmo **OMAR**, o puede ser implementado directamente si se presenta la situación donde el volumen de datos es elevado y a su vez ocurren repeticiones dentro de las observaciones, de manera tal que en resumen, la matriz de distancia o disimilaridad con la que se cuente presente entradas (fuera de la diagonal) iguales a cero.

Como aplicación de ambas técnicas se propone la base de datos de estudiantes de la Facultad de Ciencias Económicas y de Administración (FCEA) que ingresaron en el primer período de 2012 a las carreras de Contador Público, Licenciatura en Economía y Licenciatura en Administración, bajo el nuevo Plan 2012. La fuente de los registros son el Sistema General de Bedelías (SGB) y el formulario de ingreso a la FCEA de la División Estadística de la Dirección General de Planeamiento (DGPLAN).

1.1. Objetivo

El objetivo de este trabajo es generar una técnica de segmentación aglomerativa jerárquica, que dadas ciertas condiciones de los datos permite reducir el costo computacional mediante la disminución de dimensiones de la matriz de distancias que se utiliza para su desarrollo.

Se propone un método para el tratamiento de datos longitudinales tratados como datos categóricos. A su vez, se implementa una técnica que posibilita la clusterización de un conjunto de datos cuyas características principales refieren a su gran volumen y alta repetición de observaciones.

1.2. Antecedentes

Los algoritmos propuestos surgen como motivación a partir del trabajo *"Análisis logitudinal del Registro Nacional de Alumnos"de Karina Videgain, Instituto Nacional para la Evalución de la Educación, México* [Videgain, 2015], donde se analizan datos secuenciales del tipo de trayectorias, mediante la implementación de **OMA Clásico**, seguido por la aplicación de segmentación aglomerativa jerárquica.

En dicho proyecto resultó inviable trabajar con la totalidad de las observaciones por la magnitud que implicaban, por lo que se propuso implementar dichas técnicas en una muestra de los datos, logrando así reducir costos y obteniendo una implementación más manejable computacionalmente. Con el algoritmo que se propone en este proyecto, se podría haber analizado la base completa, sin necesidad de trabajar con una muestra.

1.3. Estructura del trabajo

En el primer capítulo se presenta el método de alineación de secuencias Optimal Matching Analysis (**OMA Clásico**) y su adaptativo **OMAR** (Optimal Matching Analysis Reducido), que permite generar distintos niveles de diferencias a partir de observaciones repetidas.

En el segundo capítulo se introduce una alternativa de procedimiento de clusterización aglomerativa jerárquica, cuyo principal objetivo es el ahorro computacional.

En el tercer capítulo, como aplicación se presentan resultados de los algoritmos propuestos en un set de datos de trayectorias estudiantiles de la Universidad de la República, Facultad de Ciencias Económicas y de Administración.

Finalmente se exponen las principales conclusiones de los métodos propuestos así como sugerencias de trabajos a realizar en el futuro.

Capítulo 2

Optimal Matching Analysis (OMA Clásico)

En el presente capítulo se introduce el método de alineación de secuencias Optimal Matching Analysis (**OMA Clásico**), algoritmo del que se parte para presentar un adaptativo llamado **OMAR** (Optimal Matching Analysis Reducido), que permite generar disimilaridades a partir de datos secuenciales que presentan observaciones repetidas. Se introduce el concepto de distancia para datos de tipo cualitativo, y una métrica para cuantificar dicha disimilaridad en función de la distancia de edición planteada por Levenshtein [Levenshtein, 1966] basada en operadores de edición. Dentro del método de **OMA Clásico**, se desarrolla el procedimiento que utiliza la distancia de Levenshtein, mediante el algoritmo propuesto por Needelman y Wunsch [Needleman y Wunsch, 1970]. Se implementa el método clásico de alineación de secuencias para determinar una cuantificación de las disimilaridades que se presentan, analizando frecuencias de transiciones entre los diferentes estados existentes en las distintas cadenas y asignando costos por alinear un estado con otro.

El método adaptativo propuesto, se basa entonces en la construcción de las disimilitudes entre observaciones al igual que el **OMA Clásico**, partiendo del supuesto de que dos cadenas (observaciones) que son exactamente iguales, presentan una disimilaridad igual a cero. Esta característica de los datos, permite generar una propuesta de la resolución del algoritmo clásico, de forma reducida, brindando la posibilidad de manipular volúmenes de datos mayores a un costo computacional menor.

Por último se presenta un ejemplo práctico e ilustrativo de implementación de **OMAR**, cuyo resultado es una matriz de disimilaridad reducida.

2.1. Alineación de secuencias

La técnica de Optimal Matching Analysis se compone de un conjunto de métodos para medir disimilaridades entre dos secuencias, que derivan originalmente del ámbito de la teoría de la información, y tiene como antecedente en 1950 el trabajo de Richard Hamming y en 1966 el trabajo de Vladimir Levenshtein entre otros.

Hamming [Hamming, 1950] trabajó los conceptos fundamentales de detección y corrección de códigos binarios y propuso una distancia que lleva su nombre, que trata sobre la alineación de cadenas con igual longitud, basándose únicamente en el operador de sustitución para transformar una secuencia en otra.

Años más tarde, Vladimir Levenshtein [Levenshtein, 1966] trabaja en la teoría de la codificación, que refiere a la investigación de la recepción de información codificada a través de diversos canales como radios o telégrafos con componentes de ruido y fallas en la transmisión de datos. Su propuesta implicaba contar la cantidad de ediciones necesarias para transformar una cadena en otra como en el caso de Hamming, pero no necesariamente utilizando cadenas de igual longitud. Además Levenshtein incorpora otros operadores de transformación, inserción y eliminación de caracteres [Lesnard, 2006].

En la época de los 70, Needleman y Wunsch [Needleman y Wunsch, 1970] generalizan la distancia de Levenshtein y proponen el algoritmo clásico de **OMA**. En la década del 80, el análisis de secuencias como trayectorias es introducido por Andrew Abbott y Alexandra Hrycak [Abbott y Hrycak, 1990], basándose en el trabajo de Joseph Kruskal [Kruskal, 1983] que venía investigando el algoritmo de Needleman y Wunnsch [Needleman y Wunsch, 1970] entre otros métodos de análisis de secuencias.

2.2. Introducción OMA Clásico

Una secuencia es un objeto lineal que tiene una estructura de ordenamiento unidimensional. Su análisis tiene como propósito el carácter explicativo y descriptivo de las secuencias, orientado a la generación de tipologías a partir de datos.

Una de las áreas donde más se ha extendido el uso de las técnicas de alineación de secuencias es en las Ciencias Sociales, donde generalmente el tiempo es una de las dimensiones de los datos considerada, siendo las secuencias de tipo longitudinal.

Alinear secuencias mediante una técnica de programación dinámica genera niveles de medición de alineamiento entre secuencias, que luego se emplean como insumo para algoritmos tales como segmentación, scaling o de agrupamiento, con el objetivo de encontrar patrones típicos de secuencias.

2.3. Distancias / Disimilaridades

La disimilaridad [1] es una valoración cuantitativa del nivel de alineaciones entre dos secuencias. Dicho nivel brinda una medida de la semejanza o disimilaridad entre ambas secuencias con el fin de compararlas [Studer y Ritschard, 2016].

La medida de disimilaridad a emplear depende de la técnica que se pretenda implementar y de la naturaleza de los datos disponibles, es decir, si se trata de variables cuantitativas o cualitativas. Por otro lado, el análisis de una secuencia de datos puede ser ponderado por la importancia del orden, dentro de la secuencia y en el tiempo, que presenten los posibles estados.

Cuando se trabaja con datos cuanitativos una posible técnica para medir, es la Distancia Euclídea, mientras que para datos cualitativos una de las técnicas

[1] La disimilaridad como diferencia, para considerarse una distancia debe cumplir con la propiedad triangular. En el presenta trabajo se emplean los términos disimilaridad y distancia de forma indistinta, aunque cabe recordar que las disimilaridades no siempre cumplen con la propiedad de desigualdad triangular mencionada.

utilizadas es mediante Distancias de Edición.

2.3.1. Distancia Euclídea

En un espacio euclidiano [Kaufman, 2008] de n coordenadas reales, la Distancia Euclídea entre dos puntos x y y, se define como:

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Dado que en general las secuencias obtenidas a partir de variables cualitativas no tienen coordenadas reales, no se puede emplear la Distancia Euclídea. Por eso se propone medir la distancia entre vectores utilizando técnicas basadas en Distancia de Edición.

2.3.2. Distancia de Edición

La Distancia de Edición es un algoritmo utilizado para cuantificar cuán disímiles son dos cadenas o secuencias, operando con la mínima cantidad de transformaciones. Existen varios métodos para establecer dichas distancias que varían esencialmente en el conjunto de operadores que utilizan. Éstas pueden ser definidas como un cálculo métrico parametrizable con un conjunto específico de operadores de edición, y a su vez, se puede asignar distintos costos a cada operador. Los operadores se definen en forma genérica como Insertar, Eliminar y Sustituir.

Insertar	Sea una cadena uv, al insertar el símbolo x se obtiene uxv, donde $\epsilon \longrightarrow x$ (ϵ se transforma en x, siendo ϵ el caracter nulo)
Eliminar	Sea una cadena uxv al eliminar el símbolo x se obtiene uv, donde $x \longrightarrow \epsilon$ (x se transforma en ϵ)
Sustituir	Sea una cadena uxv, donde $x \neq y$, realizando la sustitución de x en y, se produce uyv donde $x \longrightarrow y$.

Tabla 2.1: Operadores.

Las funciones de distancia deben cumplir con determinadas condiciones

para ser una métrica, definiendo una distancia entre cada par de elementos de un conjunto. Por lo que una métrica en un conjunto X, es una función de distancia $d : X \times X \longrightarrow [0, \infty)$, donde $[0, \infty)$ es un conjunto de reales no negativos y para todo $x, y, z \in X$, se cumple que:

- $d(x, y) = 0$ si y solo si $x = y$.
- $d(x, y) > 0$ si y solo si $x \neq y$.
- $d(x, y) = d(y, x)$
- $d(x, z) \leq d(x, y) + d(y, z)$. Desigualdad triangular.

Por lo tanto, la Distancia de Edición con costos no negativos, satisface los axiomas de una métrica generando un espacio métrico de cadenas. Para esto se asume que todas las operaciones tienen costos positivos y que para cada una de ellas existe una operación inversa con igual costo.

En el presente trabajo se utiliza el algoritmo de Optimal Matching Analysis Clásico [Lesnard, 2006] para alinear secuencias, mediante la Distancia de Edición de Levenshtein. Se toma como referencia el algoritmo Optimal Matching Analysis propuesto en el paquete *TraMinerR* [Gabadinho *et al.*, 2009] del software libre R [Team, 2019] para la visualización de secuencias categóricas de trayectorias. Dentro de las distintas variantes de análisis de trayectorias evaluadas por *TraMinerR*, se desarrolló el método de **OMA Clásico** basado en la disimilaridad de Levenshtein [Levenshtein, 1966] con costo de *indel* (Insertar/Eliminar) constante en 1.

2.3.2.1. Distancia de Levenshtein

Se define $X = \{c_1, c_2, \ldots, c_k\}$ un alfabeto de elementos finito-numerables. Sea a una cadena definida como una secuencia finita de caracteres $a = (c_1, c_2, \ldots, c_m)$, con $c_k \in X$, $k = 1, \ldots, m$.

Sea la cadena a sobre el alfabeto X finito y c_k es el k-ésimo elemento (caracter) de a.

Se define a la Distancia de Levenshtein [Levenshtein, 1966] entre dos cadenas a y b, de largo $|a|$ y $|b|$ respectivamente, como el resultado de aplicar la siguiente fórmula recursiva:

$$d_{a,b}(i,j) = \begin{cases} \text{máx}(i,j) & \text{mín}(i,j) = 0 \\ \text{mín} \begin{cases} d_{a,b}(i-1,j)+1 & Insertar \\ d_{a,b}(i,j-1)+1 & Eliminar \\ d_{a,b}(i-1,j-1)+\mathbb{I}_{(a_i \neq b_j)} & Sustituir \end{cases} & \text{mín}(i,j) > 0 \end{cases}$$

donde i es la i-ésima fila y j es la j-ésima columna, con $i \in \{0, 1, ..., |a|\}$, $j \in \{0, 1, ..., |b|\}$. La cadena a comienza en la posición $i = 1$ y la cadena b comienza en la posición $j = 1$. $\mathbb{I}_{(a_i \neq b_j)}$ es la función indicadora que es 0 cuando $a_i = b_j$.

Por tanto, la Distancia de Edición entre dos cadenas es el mínimo número de ediciones necesarias para transformar una cadena en otra.

Se pueden definir algunas características para la Distancia de Edición (Levenshtein):

- Es al menos la diferencia de longitud de las dos cadenas.
- Es como máximo el largo de la cadena más grande.
- Es cero si y sólo si las cadenas son iguales.
- Si las cadenas son de igual longitud se trata de la distancia de Hamming[1].

2.4. OMA Clásico

El Método Optimal Matching Analysis **(OMA Clásico)** se basa en el algoritmo de Needelman y Wunsch [Anexo: A]. Fue introducido por Andrew Abbott y Alexandra Hrycak [Abbott y Hrycak, 1990] en el área de Ciencias Sociales, aunque dicha técnica fue utilizada previamente en el área biológica en la alineación de cadenas de proteínas. Como antecedente al **OMA Clásico** se puede considerar la distancia de Levenshtein [Levenshtein, 1966], que utiliza

[1] **Distancia de Hamming**: La distancia de Edición de Hamming [Hamming, 1950] es un caso particular de la distancia de Levenshtein, donde solo se consideran las diferencias entre cadenas de igual tamaño, utilizando únicamente el operador de sustitución. La distancia entre cadenas es el número de caracteres que sean distintos en igual posición. Es decir, es una medida del mínimo número de operaciones requeridas para transformar una cadena en otra: más cerca de cero, más parecidas son las cadenas.

como sistema de *"score"* las operaciones básicas con igual peso, es decir con costo 1.[1]

El **OMA Clásico** es un método que se aplica a los datos ordenados en secuencias, con repeticiones de eventos potenciales. Se utiliza para identificar patrones en las trayectorias en un período determinado. El grado de disimilaridad entre dos cadenas es calculado por el número de operaciones de edición que son necesarias para transformar una secuencia en otra. Se utilizan tres operaciones básicas: *inserción*, *eliminación* y *sustitución*.

En este método se consideran secuencias que tiene definidos sus posibles estados en un alfabeto finito, donde cada posición en la cadena indica un determinado orden de los estados. En tiempo discreto, el estado de una secuencia en una posición t brinda información de tiempo.La cantidad de posiciones de una cadena determinan el período de tiempo considerado. Por lo que la diferencia entre dos posiciones definen una duración.

Además, este algoritmo es una técnica de optimización, donde se deben considerar todas las posibles combinaciones de edición para alinear dos secuencias con el fin de identificar la solución más eficiente. Para k categorías de estados y un período de tiempo de largo t, se tienen $\sum_{i=1}^{t} k^i$ posibles secuencias diferentes. Se deben definir los costos necesarios para alinear estado a estado (caracter a caracter) cuando se transforma una cadena en otra. Cada transformación posible implica un costo ya sea de *sustitución*, *eliminación* o *inserción*.

Por lo tanto, el **OMA Clásico** define la disimilaridad como el mínimo de la agregación de costos (*sustitución* e *indels*) necesarios para transformar una cadena en otra. Para esto, utiliza costo *indel = 1* y costos de *sustitución* basados en las frecuencias de transición de cada estado que presentan los datos.

[1]El **OMA Clásico**, además de la distancia de Levenshtein con costo *indel* 1, utiliza el costo de *sustitución* basado en las frecuencias de transición de los estados observados.

2.4.1. Matriz de Costos de Sustitución, Matriz de Transición y Costo Indel

Sea **W** una matriz de **Costos de Sustitución** de $(|X|+1)\times(|X|+1)$. Los costos deben definir una métrica entre los estados posibles. La matriz **W** debe ser simétrica, satisfacer la desigualdad triangular y ser cero para la sustitución de un elemento por sí mismo. Los **Costos de Sustitución** se basan en la **Matriz de Transición**, que describe las trayectorias entre todos los eventos posibles en un cierto momento t. Las transiciones son los vínculos que existen entre los distintos estados; una transición baja indica que dichos estados en el tiempo t casi no se comunican.
Se asigna mayor costo cuando la transición es rara, y bajo costo cuando es más frecuente.

La tasa de transición entre dos estados a y b, es considerada como la probabilidad $p(b|a)$ de pasar del estado a al estado b en posiciones sucesivas.
Asumiendo invarianza en el tiempo, la frecuencia de transición es estimada como:

$$p(b|a) \;=\; \frac{\sum_{t=1}^{L-1} n_{t,t+1}(a,b)}{\sum_{t=1}^{L-1} n_t(a)}$$

donde L es el largo máximo de la secuencia observada, $n_t(a)$ es la cantidad de estados a en la secuencia en la posición t, y $n_{t,t+1}(a,b)$ es la cantidad de estados a en la posición en tiempo t y con el estado b en la posición en $t+1$.

El **Costo de Sustitución** se define como:

$$w(a,b) \;=\; 2 \;-\; p(a|b) \;-\; p(b|a)$$

donde $p(a|b)$ es la probabilidad de pasar del estado b al estado a en tiempos sucesivos, $p(b|a)$ es la probabilidad de pasar del estado a al estado b en tiempos sucesivos.

El **Costo Indel** surge de aplicar el operador de *inserción* y *eliminación*, siendo fijo en el algoritmo **OMA Clásico** (igual a 1) [Gabadinho *et al.*, 2009] donde se evalúa cuándo insertar un elemento extra o eliminar un elemento en una o más de las posiciones de una secuencia, reduciendo los costos en general.

2.5. Procedimiento del algoritmo OMA Clásico

Es un método recursivo de programación dinámica [Abbott y Hrycak, 1990]. Se basa en tres posibles direcciones (izquierda, arriba y diagonal) que pueden tomar las operaciones de edición en una matriz de resolución.

La matriz de resolución es una herramienta auxiliar que permite entender el proceso del algoritmo. Es de dimensión $(m+1) \times (n+1)$ donde m y n son los largos de cada secuencia comparada.

Las transformaciones necesarias para alinear dos secuencias, son representadas en dicha matriz recorriendo las celdas con movimientos horizontales, verticales y diagonales. Un movimiento horizontal representa la inserción de un elemento, un movimiento vertical representa la eliminación de un elemento, mientras que un movimiento diagonal es el costo por moverse libremente si los elementos de la fila y de la columna coinciden, o directamente por realizar una sustitución.

Cada celda tiene la disimilaridad mínima acumulada. La celda superior izquierda toma valor cero, y el resto de la primera columna y fila se completan incrementando por el costo *indel*.

Las celdas restantes $C_{i,j}$ se completan calculando las disimilaridades parciales con:

$$C_{i,j} = \text{mín} \begin{cases} c_{i-1,j-1} + w_{i,j} \\ c_{i-1,j} + d \\ c_{i,j-1} + d \end{cases}$$

donde i es el i-ésimo elemento de la cadena a y j el j-ésimo elemento de la cadena b, d es el costo de *indel*, $w_{i,j}$ es el costo de sustitución y $c_{i,j}$ es la disimilaridad parcial calculada en la celda $C_{i,j}$.

El algoritmo calcula cuál de las tres operaciones es más barata, hasta completar todas las celdas y llegar a la celda inferior derecha, siendo el valor de

esta última, el costo del camino minimal para transformar una secuencia en otra.

2.6. Algoritmo OMA Reducido Propuesto

El OMA Reducido **(OMAR)** propone implementar el algoritmo **OMA Clásico** disminuyendo la dimensión de la matriz de disimilaridad. De esta forma se utilizan solamente *secuencias únicas*, es decir que todas aquellas que tienen distancia cero respecto a las únicas, son descartadas.

Se identifica como *secuencias únicas* a la primera aparición de una secuencia en la matriz de datos, siendo las repetidas las siguientes apariciones de dicha secuencia. Es así que se obtiene una matriz de disimilaridad de menor tamaño, pasando de $n \times n$ a $k \times k$ con $k \leq n$, donde n es la cantidad de observaciones y k la cantidad de observaciones únicas.

El método **OMA Clásico** calcula las disimilaridades entre todos los pares de secuencias con el fin de generar la matriz de disimilaridad de resultado. Dicho cálculo puede ser parcialmente sustituido por el algoritmo **OMAR**, pues se alinean solamente los pares de secuencias que son diferentes, cuyas distancias son distintas de cero.

En una primera etapa se identifican las secuencias que son iguales en la matriz de datos, almacenando solo una de ellas y contabilizando la frecuencia de las mismas, guardando el orden de aparición.

Sean T_i, T_j y T_k tres secuencias tales que $d(T_i, T_j) = 0$, $d(T_i, T_k) \neq 0$ y $d(T_j, T_k) \neq 0$.

Sea v el vector de frecuencias de las observaciones, tal que en primer instancia se tiene que $v = (1, 1, 1)$, es decir se computa la frecuencia absoluta de cada observación.

Una vez que se reduce la matriz de datos, eliminando la *secuencia repetida* T_j, se actualiza el vector v de frecuencias para computar la cantidad de repeti-

ciones, siendo $v = (2, 1)$, donde el dato T_i presenta una frecuencia absoluta de 2, pues aparece dos veces en el set de datos como T_i y como T_j (se asume que es el mismo dato debido a que la distancia entre ambas secuencias es cero).

Se repite el procedimiento $n - k$ veces, hasta guardar todas las *secuencias únicas* y eliminar todas las *secuencias repetidas*, actualizando en cada paso el vector de frecuencias, siendo éste finalmente un vector de frecuencias absolutas de datos repetidos de largo k.

Por lo tanto, el método propuesto reduce iterativamente la matriz de datos, identificando las *secuencias únicas* y actualizando el vector de frecuencias, lo que posibilita construir la matriz de disimilaridad reducida a partir solamente del cálculo de distancias entre pares de *secuencias únicas*.

Dependiendo de la naturaleza de los datos y de la cantidad de observaciones repetidas, este procedimiento puede lograr un ahorro computacional importante al tener que computar solamente $k \times k$ disimilaridades. La construcción de la matriz de disimilaridades se realiza con aquellas observaciones que son distintas, teniendo en cuenta el vector de frecuencias y el orden de aparición por primera vez de cada *secuencia única* en la matriz de datos.

2.7. Ejemplo OMA Clásico

Se ejemplifica el método **OMA Clásico** con los siguientes datos secuenciales en 4 períodos t, donde T_k, $k = 1, ..., 7$ son secuencias de igual tamaño, definidas a partir de un alfabeto de 4 caracteres.

Sea X el alfabeto tal que $X = \{A, H, L, O\}$ con estados A, H, L, O.

Se presentan la matriz de datos según períodos, la matriz de transiciones de estados y la matriz de costos de sustitución calculados a partir de las frecuencias de transición.

	Período 1	Período 2	Período 3	Período 4
T_1	H	O	L	A
T_2	L	O	L	A
T_3	H	O	L	A
T_4	H	O	L	A
T_5	A	A	H	O
T_6	H	O	L	A
T_7	A	A	H	O

Tabla 2.2: Matriz de datos secuenciales.

Las frecuencias son consideradas independientes de la posición de los estados e invariantes en el tiempo. En cada fila de la matriz de transición se obtiene la distribución de transiciones desde el estado en tiempo t al estado en $t + 1$, sumando cada fila 1.

	$>> A$	$>> H$	$>> L$	$>> O$
$A >>$	0.50	0.50	0.00	0.00
$H >>$	0.00	0.00	0.00	1.00
$L >>$	0.83	0.00	0.00	0.17
$O >>$	0.00	0.00	1.00	0.00

Tabla 2.3: Matriz de transición.

En el ejemplo el estado L aparece 6 veces, se pasa del estado L al estado O, 1 vez y 5 veces al estado A, por lo que la probabilidad de ir del estado L al estado A es $\frac{5}{6}$ y pasar del estado L al estado O es $\frac{1}{6}$.

La matriz de costos de sustitución **W** se calcula a partir de la matriz de transición, considerando las frecuencias de cada estado.

	A	H	L	O
A	0.00	1.50	1.17	2.00
H	1.50	0.00	2.00	1.00
L	1.17	2.00	0.00	0.83
O	2.00	1.00	0.83	0.00

Tabla 2.4: Matriz de costos de sustitución **W**.

El mínimo costo de sustitución posible es cero, es decir alinear un estado consigo mismo (pasar del estado A al estado A). Por otra parte, el máximo costo de sustitución posible es 2 cuando no se observa la transición de un estado a otro, por ejemplo pasar del estado A al O.

A partir de los costos de sustitución w calculados y con costo de *indel* $d = 1$, se alinean las cadenas $LOLA$ y $HOLA$ como ejemplo, empleando el algoritmo **OMA Clásico**. La matriz de resolución es la que se muestra en la figura 2.1 donde se obtiene una disimilaridad igual a 2.

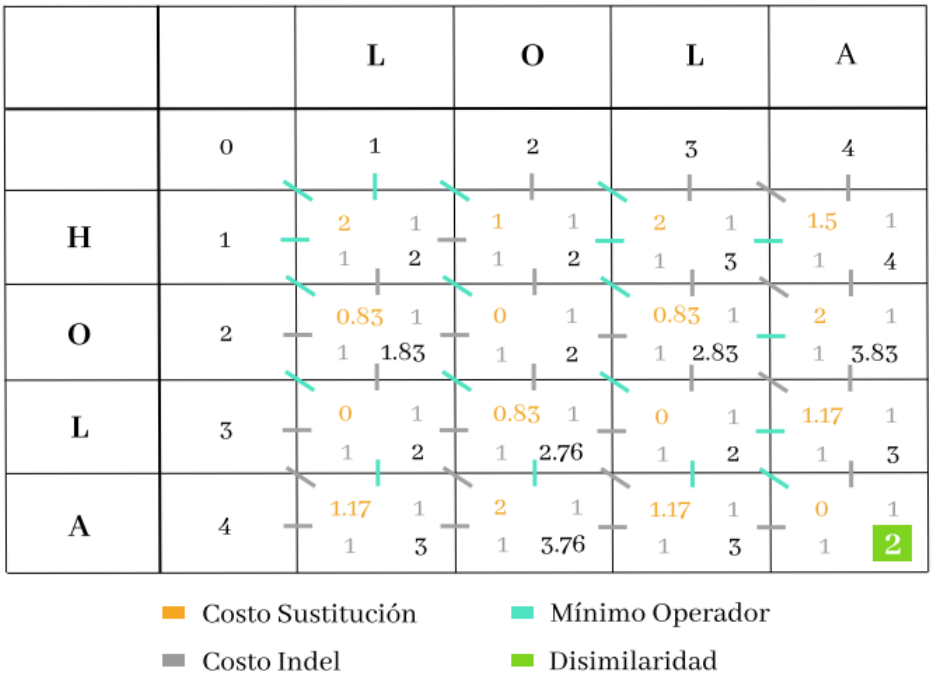

Figura 2.1: Alineamiento de cadenas HOLA y LOLA.

Una vez que se han alineado todos los pares de secuencias obteniendo sus respectivas disimilaridades, se construye la matriz de disimilaridades computando las mismas en las celdas correspondientes a cada par de secuencias.

	T_1	T_2	T_3	T_4	T_5	T_6	T_7
T_1	0	**2**	0	0	4	0	4
T_2	**2**	0	**2**	**2**	5	**2**	5
T_3	0	**2**	0	0	4	0	4
T_4	0	**2**	0	0	4	0	4
T_5	4	5	4	4	0	4	0
T_6	0	**2**	0	0	4	0	4
T_7	4	5	4	4	0	4	0

Tabla 2.5: Matriz de disimilaridades.

2.7.1. ¿Cómo se completa la matriz de resolución?

La matriz de resolución es un proceso de alineación que ayuda a entender cómo funciona el algoritmo. El **OMA Clásico** está planteado como un problema de optimización donde se deben evaluar todas las posibles combinaciones de operadores de edición para alinear dos secuencias de forma de encontrar la solución más eficiente. El problema se resuelve recursivamente mediante programación dinámica basándose en las tres formas de llegar a una celda de la matriz, desde la izquierda, desde arriba y desde la diagonal. Cada una de estas tres direcciones significa uno de los operadores de edición. Un movimiento horizontal representa una inserción para alinear un estado de una secuencia con la otra. Un movimiento vertical corresponde a una eliminación. Un movimiento en diagonal puede significar un movimiento sin costo cuando los estados son iguales o para implementar una sustitución de un estado por otro. Cada celda va acumulando el mínimo costo alcanzado, o sea la disimilaridad parcial entre las cadenas. Al llegar a la celda inferior derecha, se alcanza la solución minimal del proceso obteniendo la disimilaridad entre las dos cadenas [Lesnard, 2006].

Continuando con el ejemplo, para completar la matriz de resolución de dimensión $(m+1) \times (n+1)$, con m y n los largos de cada secuencia, sean:

- Secuencia 1 (T_1): $HOLA$ de tamaño $n = 4$.
- Secuencia 2 (T_2): $LOLA$ de tamaño $m = 4$.

A partir de la fórmula recursiva de Levenshtein aplicando la primera parte de la función se completa la celda superior izquierda con el valor cero (se considera 0 como el primer índice de la matriz) y la primer fila y columna incrementando de a uno (el máximo índice de la fila/columna cuando el índice de la columna/fila es cero), por ser el costo *indel* igual a 1 [Fórmula de Levenshtein en sección: 2.3.2.1].

		L	O	L	A
	0	1	2	3	4
H	1				
O	2				
L	3				
A	4				

Tabla 2.6: Matriz de resolución. Paso 1.

Para completar la celda (1,1) que corresponde a la disimilaridad parcial entre el las secuencias T_1 y T_2 donde se comparan los estados L y H en el tiempo 1:

$$d_L = d_{T_1,T_2}(1,1) = \begin{cases} \text{máx}(1,1) & \text{mín}(1,1) = 0 \\ \text{mín} \begin{cases} d_{T_1,T_2}(0,1) + 1 & Insertar \\ d_{T_1,T_2}(1,0) + 1 & Eliminar \\ d_{T_1,T_2}(0,0) + 2 & Sustituir \end{cases} \end{cases}$$

Siendo 2 el costo de sustitución de pasar del estado L al estado H y 1 el costo *indel*.

$$d_L = d_{T_1,T_2}(1,1) = \begin{cases} \text{máx}(1,1) & \text{mín}(1,1) = 0 \\ \text{mín} \begin{cases} 1+1=2 & Insertar \\ 1+1=2 & Eliminar \\ 1+2=3 & Sustituir \end{cases} \end{cases}$$

La disimilaridad parcial es el mínimo resultado de $d_{T_1,T_2}(1,1) = 2$, en este caso surge de operar con costo *indel*.

		L	O	L	A
	0	1	2	3	4
H	1	2			
O	2				
L	3				
A	4				

Tabla 2.7: Matriz de resolución. Paso 2.

Completando la celda (2,1):

$$d_L = d_{T_1,T_2}(2,1) = \begin{cases} \text{máx}(2,1) & \text{mín}(2,1) = 0 \\ \text{mín} \begin{cases} d_{T_1,T_2}(1,1)+1 & Insertar \\ d_{T_1,T_2}(2,0)+1 & Eliminar \\ d_{T_1,T_2}(1,0)+0.83 & Sustituir \end{cases} \end{cases}$$

Siendo 0.83 el costo de sustitución de pasar del estado O al estado L y 1 el costo *indel*.

$$d_L = d_{T_1,T_2}(2,1) = \begin{cases} \text{máx}(2,1) & \text{mín}(2,1) = 0 \\ \text{mín} \begin{cases} 2+1=3 & Insertar \\ 2+1=3 & Eliminar \\ 1+0.83=1.83 & Sustituir \end{cases} \end{cases}$$

La disimilaridad parcial es el mínimo resultado de $d_{T_1,T_2}(2,1) = 1.83$, que surge por operar con el mínimo costo, en este caso de sustitución.

		L	O	L	A
	0	1	2	3	4
H	1	2			
O	2	1.83			
L	3				
A	4				

Tabla 2.8: Matriz de resolución. Paso 3.

La matriz de resolución completa es la siguiente:

		L	O	L	A
	0	1	2	3	4
H	1	2	2	3	4
O	2	1.83	2	2.83	3.83
L	3	2	2.76	2	3
A	4	3	3.76	3	2

Tabla 2.9: Matriz de resolución. Paso final.

El resultado final de alinear las secuencias *HOLA* y *LOLA* es de disimilaridad 2.

2.8. Ejemplo OMAR Propuesto

El algoritmo **OMAR** propuesto busca identificar aquellas secuencias que son únicas y descartar sus repeticiones. Continuando con el ejemplo anterior se identifica a las secuencias T_1, T_2 y T_5 como *secuencias únicas*, y las restantes como repeticiones de las mismas.

Se observa por ejemplo, que las secuencias T_1, T_3, T_4 y T_6 son iguales, determinando como *secuencia única* a T_1, siendo T_3, T_4 y T_6 repeticiones de esta.

Asimismo se almacena tanto la frecuencia de dicha secuencia en el vector v, como el lugar de la primer aparición en la matriz de datos en el vector posición p, donde $v = 4$ y $p = 1$.

Una vez determinadas las *secuencias únicas* y sus repeticiones, se eliminan las *secuencias repetidas* reteniendo las *secuencias únicas*.

En este caso, se retienen las secuencias T_1, T_2 y T_5 y se eliminan las secuencias T_3, T_4, T_6 y T_7. El vector de frecuencias es $v = (v_1, v_2, v_5) = (4, 1, 2)$ y el vector de posiciones es $p = (p_1, p_2, p_5) = (1, 2, 5)$.

	Período 1	Período 2	Período 3	Período 4
T_1	H	O	L	A
T_2	L	O	L	A
T_5	A	A	H	O

Tabla 2.10: Matriz de *secuencias únicas*.

Luego de realizada la eliminación de *secuencias repetidas*, se calcula mediante **OMA Clásico** con costos de sustitución basados en frecuencias de transición e *indel* igual a 1, solamente las disimilaridades dos a dos de las *secuencias únicas*. Se reduce así la cantidad de cálculos de $(n^2-n)/2$ a $(k^2-k)/2$ donde n es la cantidad de observaciones y k es la cantidad de *secuencias únicas*, en este caso se deben realizar 3 cálculos de disimilaridades en lugar de 21.

	T_1	T_2	T_5
T_1	0	2	4
T_2	2	0	5
T_5	4	5	0

Tabla 2.11: Matriz de disimilaridad de *secuencias únicas*.

Por último, con las disimilaridades calculadas con secuencias únicas, el vector de frecuencias y el vector de posiciones, es posible reconstruir la matriz de disimilaridades de $n \times n$, sin realizar ningún cálculo adicional.

Capítulo 3

Análisis de Cluster

En el presente capítulo se introduce el procedimiento de clusterización aglomerativa jerárquica, que dadas ciertas características de los datos, permitirá la propuesta de un algoritmo de segmentación adaptativo cuyo principal objetivo es el ahorro computacional. Dentro de los métodos de clusterización, se desarrolla el algoritmo **Agnes**, propuesto por Kaufman [Kaufman, 2008] y se presenta la fórmula de *Lance y Williams* [Lance y Williams, 1966] para la actualización de las disimilaridades en cada paso iterativo de la clusterización. Se desarrolla dicha formulación recursiva para distintos criterios con los que se definen posibles algoritmos como *Vecinos más cercanos, Vecinos más lejanos, Average, Weigthed* y *Ward*.

En la siguiente parte del capítulo se discute sobre las limitaciones y características posibles, principalmente encontradas en datos de tipo cualitativo. Se presenta entonces un caso de estudio donde existe un número considerable de medidas de distancias iguales a cero, donde la característica del set de datos presenta observaciones con valores repetidos en sus distintas variables.

Las últimas secciones del capítulo consisten en el desarrollo de una propuesta para implementar una adaptación del algoritmo **Agnes**, llamada **AGNES-R** (Agnes Reducido) considerando la existencia de distancias que sean cero entre observaciones. En el procedimiento **AGNES-R** se desarrollan los distintos criterios para determinar las distancias a partir de la formulación de actualización de *Lance y Williams*. Se presenta un ejemplo práctico e ilustrativo de la implementación de **AGNES-R**.

3.1. Introducción

El análisis de cluster es una técnica multivariada, que tiene como finalidad agrupar elementos en clusters homogéneos, en base a las posibles similitudes o diferencias existentes entre los individuos que componen un conjunto de datos.

Dicha técnica busca que los elementos pertenecientes a un mismo cluster sean lo más homogéneos posible entre sí, mientras que los grupos formados sean lo más heterogéneos entre sí.

El análisis de cluster en sí mismo no es un algoritmo específico, sino la forma general de resolver un problema. Existe una amplia variedad de métodos para realizar dicha segmentación, los que esencialmente dependen de la naturaleza y cantidad de datos considerados. En consecuencia, al implementar dichas técnicas las soluciones pueden no ser únicas, pues dependen del procedimiento y método seleccionado.

El propósito de la construcción de clusters es particionar al conjunto de datos en partes más pequeñas e identificar posibles patrones existentes.

El proceso implica segmentar el conjunto de datos de forma tal que cada individuo pertenezca a un único cluster, y a su vez que todos los clusters contengan a todos los individuos; es una partición exhaustiva y de grupos mutuamente excluyentes.

3.2. Procedimiento del algoritmo aglomerativo jerárquico

El algoritmo consiste en una serie de particiones que se ejecuta a partir de considerar tantos grupos como individuos existen en estudio, los cuales se van aglomerando hasta llegar a conformar un único cluster que contiene todas las observaciones. Este procedimiento se caracteriza por no definir previamente la cantidad de clusters a obtener.

El algoritmo parte entonces de una matriz de distancias o disimilaridades

entre los elementos, y en base al procedimento seleccionado de aglomeración se construye una jerarquía.

Dadas n observaciones, el procedimiento permite reducir a $n-1$ conjuntos mutuamente excluyentes, considerando todas las posibles $n(n-1)/2$ uniones de pares y seleccionando aquella unión que maximiza alguna función objetivo[1] elegida previamente.

En la figura 3.1 se presenta un ejemplo de procedimiento de estructura jerárquica al implementar una clusrerización aglomerativa jerárquica, sobre el conjunto de datos de obseraciones T_1, T_2,..., T_7 de la sección: 2.7.

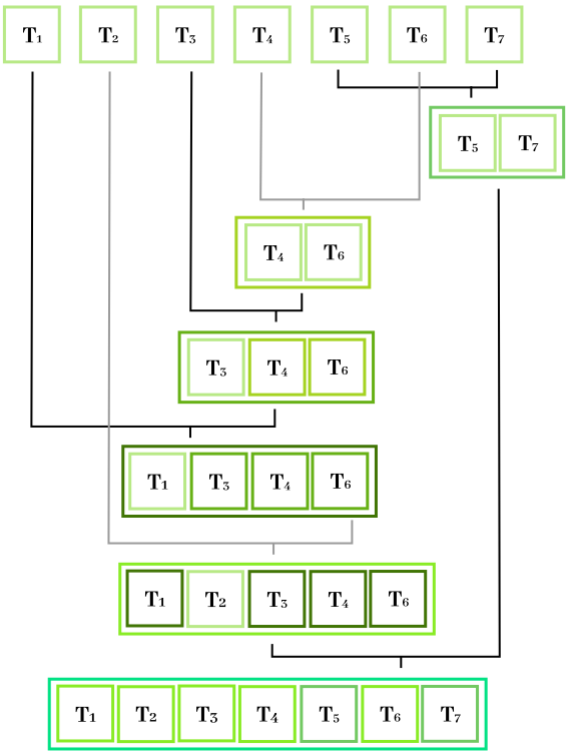

Figura 3.1: Estructura jerárquica.

Repitiendo este procedimiento se obtienen estructuras jerárquicas y se puede calcular en cada etapa de agrupamiento una estimación cuantitativa de la pérdida de homogeneidad asociada.

[1]Función Objetivo: En general se utiliza la suma de los errores al cuadrado.

3.2.1. Método aglomerativo jerárquico: Agnes

Dentro de los distintosla métodos existentes de la aglomeración jerárquica, el presente trabajo se centra en la técnica **Agnes** (Agglomerative Nesting) propuesta por Kaufman [Kaufman, 2008].

Al comienzo del algoritmo se considera que todas las observaciones son clusters de un único elemento. En la siguiente etapa las observaciones más cercanas o similares son agrupadas formando un nuevo cluster de dos observaciones, quedando el resto de los clusters invariantes. En la matriz de disimilaridad se busca, la entrada de menor valor entre observaciones, o sea aquellas dos observaciones más similares, para posteriormente realizar la unión de las mismas. En caso que exista más de un par de observaciones con disimilaridad minimal, se selecciona a uno de estos pares de forma aleatoria.

En sucesivas etapas, se irán juntando clusters con una única observación o clusters con más de una observación. Por ello, en cada paso iterativo de aglomeración no se debe definir únicamente la disimilaridad inicial entre las observaciones, sino también la disimilaridad entre dos clusters cualesquiera.

Para medir dichas disimilaridades se proponen cinco criterios utilizados en cluster aglomerativo jerárquico, mediante la fórmula recursiva propuesta por *Lance y Williams* [Lance y Williams, 1966].

3.3. Recursión de Lance y Williams

La fórmula de *Lance y Williams* (L-W) [Lance y Williams, 1966] proporciona una expresión para actualizar las disimilaridades (o distancias) entre clusters a medida que los mismos se van aglomerando. Los parámetros utilizados en la actualización dependen de la cardinalidad de cada cluster y de las disimilaridades (distancias) de los grupos que se aglomeran en cada etapa.

Sean R y Q los clusters, tal que $R = A \cup B$, un cluster formado en una etapa anterior (no necesariamente inmediatamente anterior), al agruparse los clusters A y B.

Entonces, se redefine la disimilaridad relativa respecto al cluster Q, tal que:

$$\begin{aligned} d(R,Q) &= d((A,B),Q) \\ &= \alpha_A d(A,Q) + \alpha_B d(B,Q) + \beta d(A,B) + \gamma |d(A,Q) - d(B,Q)| \end{aligned} \tag{3.1}$$

donde d es la disimilaridad (o distancia) utilizada, α_A, α_B, β y γ coeficientes que se definen según el criterio de clusterización que se elija y $|\ .\ |$ denota el valor absoluto, A y B los elementos que se fusionan en el cluster R.

Se asume que los tres grupos Q, A y B que contienen $|Q|$, $|A|$ y $|B|$ elementos respectivamente y que las distancias entre grupos $d(A,Q)$, $d(B,Q)$ y $d(A,B)$ fueron anteriormente definidas. Se supone que en el paso jerárquico analizado la $d(A,B)$ es la mínima existente, por lo que el grupo A y el grupo B se fusionan formando un nuevo cluster R, con $|R|$ igual a $|A| + |B|$ elementos. Por lo tanto, en cada paso iterativo es necesario tener definida la cardinalidad de cada grupo así como las distancias de los grupos en cuestion.

3.4. Criterios de actualización de disimilaridades según Lance y Williams

La formulación de *Lance y Williams* establece la actualización de las disimilaridades a partir del momento que se fusionan dos grupos. El criterio de aglomeración seleccionado [Maechler *et al.*, 2009] establece los distintos órdenes de jerarquía que van sucediendo en el procedimiento. Por lo tanto, los parámetros utilizados en la fórmula de actualización de L-W dependerán del criterio de actualización que se determine. Se plantean 5 estrategias cuyos parámetros son:

	α_i	α_j	β	γ
VML	1/2	1/2	1/2	0
VMC	1/2	1/2	-1/2	0
$Weighted$	1/2	1/2	-1/4	0
$Average$	$\frac{\|A\|}{\|R\|}$	$\frac{\|B\|}{\|R\|}$	0	0
$Ward$	$\frac{\|A\|+\|Q\|}{\|R\|+\|Q\|}$	$\frac{\|B\|+\|Q\|}{\|R\|+\|Q\|}$	$-\frac{\|Q\|}{\|R\|+\|Q\|}$	0

Tabla 3.1: Coeficientes de L-W según criterios.

3.4.1. Vecinos más cercanos (VMC) - Simple linkage

El algoritmo fue propuesto por Florek (1951) y Sneath (1957) [Kaufman, 2008]. Considera la distancia mínima entre los pares de elementos de un cluster R, formado por A y B con los elementos del cluster Q, tal que:

$$d(R,Q) = \min_{i \in R, j \in Q} \Big\{ d(i,j) \Big\}$$

donde i es el i-ésimo elemento del cluster R y j es el j-ésimo elemento del cluster Q.

La disimilaridad por *Vecinos más cercanos* se puede computar por la expresión de recurrencia de *Lance y Williams* mediante la fórmula de actualización de disimilaridades en cada paso iterativo donde se fusionan grupos:

$$\begin{aligned} d(R,Q) &= \text{mín} \Big\{ d(A,Q),\ d(B,Q) \Big\} \\ &= \frac{1}{2} \Big(d(A,Q) \ +\ d(B,Q) \Big) \ -\ \frac{1}{2} \left| d(A,Q) \ -\ d(B,Q) \right| \end{aligned}$$

donde $\alpha_a = \alpha_b = \frac{1}{2}$ y $\beta = -\frac{1}{2}$ con $\gamma = 0$

3.4.2. Vecinos más lejanos (VML) - Complete linkage

Método propuesto por McQuitty (1960) y por Sokel and Sneath (1963) [Kaufman, 2008]. Las disimilaridades por *Vecinos más lejanos* entre dos cluster

se define como la mayor disimilaridad entre un elemento de un cluster y un elemento perteneciente a otro cluster, es decir como la máxima distancia entre un par de elementos del grupo $R/R = \{A, B\}$ y los elementos del grupo Q, tal que:

$$d(R, Q) = \max_{i \in R, j \in Q} \Big\{ d(i, j) \Big\}$$

donde i es el i-ésimo elemento del cluster R y j es el j-ésimo elemento del cluster Q.

La ecuación de actualización a través de la fórmula de *Lance y Williams* se define como:

$$\begin{aligned} d(R, Q) &= \max \Big\{ d(A, Q),\ d(B, Q) \Big\} \\ &= \frac{1}{2} \Big(d(A, Q) \ + \ d(B, Q) \Big) \ + \ \frac{1}{2} \left| d(A, Q) \ - \ d(B, Q) \right| \end{aligned}$$

donde $\alpha_a = \alpha_b = \beta = \frac{1}{2}$ con $\gamma = 0$.

Tanto el método de *Vecinos más cercanos*, como el de *Vecinos más lejanos* producen una estructura jerárquica invariante al actualizar la matriz de disimilaridades mediante el método de *Lance y Williams* (transformación monótona).

3.4.3. Average - Unweighted pair group method average

Algoritmo inicialmente propuesto por Sokly Michaener en 1958 [Kaufman, 2008], para variables escaladas por intervalo. Dados dos cluster R y Q, donde $|R|$ y $|Q|$ son los tamaños de los cluster R y Q respectivamente, la disimilaridad entre el cluster R y el cluster Q, $d(R, Q)$, se define como el promedio de todas las disimilaridades $d(i, j)$ donde $i \in R$ y $j \in Q$, es decir:

$$
\begin{aligned}
d(R,Q) &= \frac{1}{|R|\,|Q|} \sum_{j\in Q, i\in R} d(i,j) \\
&= \frac{|A|}{|R|}\,\frac{1}{|A|\,|Q|} \sum_{j\in Q, i\in A} d(i,j) \;+\; \frac{|B|}{|R|}\,\frac{1}{|B|\,|Q|} \sum_{j\in Q, i\in B} d(i,j) \qquad (3.2) \\
&= \frac{|A|}{|R|}\,d(A,Q) \;+\; \frac{|B|}{|R|}\,d(B,Q)
\end{aligned}
$$

Las disimilaridades que se utilizan para calcular $d(R,Q)$ no involucran al cluster R, por lo que éstas en todos los pasos permanecen invariantes. Solo dos disimilaridades (entre A y Q y entre B y Q) tienen que ser consideradas en lugar de examinar todas las disimilaridades entre los elementos de R y los elementos de Q.

Por lo tanto, todas las disimilaridades que no involucran al cluster R, permanecen iguales al paso donde se unieron previamente (u origen) y aquellas que sí involucran a R son computadas con la expresión de actualización de *Lance y Williams*.

3.4.4. Weighted Average Linkage - WPGMA

Algoritmo propuesto por Sokal y Sneath (1963) [Kaufman, 2008]. Es una variante del algoritmo *Average*.

Las disimilaridades se actualizan mediante la fórmula de L-W con la siguiente expresión:

$$
d(R,Q) = \frac{1}{2}\,d(A,Q) \;+\; \frac{1}{2}\,d(B,Q) \qquad (3.3)
$$

En el método *Average* (UPGAM), se ponderan todas las disimilaridades de igual manera, sin depender del tamaño de los grupos unidos (como en el criterio 3.2). Es decir, no se considera si el cluster A tiene mayor o menor cardinalidad respecto al cluster B, por lo que todas las actualizaciones de disimilaridad tienen igual peso.

En cambio, en el método *Weighted*, aunque se fusionen dos clusters uno de ellos con menor cardinalidad, tendrá en definitiva mayor peso en la actualización de la disimilaridad (ambas disimilaridades están ponderadas por 1/2), pues no intervienen la cantidad de elementos en dicha actualización.

3.4.5. Método *Ward*

El método de *Ward* (1963) [Kaufman, 2008], es un procedimiento jerárquico que se basa en unir dos clusters en cada etapa, donde el incremento de la suma de las diferencias al cuadrado, de las observaciones de cada cluster respecto al centroide del mismo, sea menor.

Se define entonces la suma de los cuadrados de los errores de un cluster como la suma de las distancias euclídeas entre los elementos del cluster C y su centroide $\overline{x}(C)$:

$$SSE = \sum_{i \in C} \parallel x_i - \overline{x}(C) \parallel^2$$

donde i es el i-ésimo elemento del cluster C.

Por lo tanto, el incremento de la suma de los errores al cuadrado ΔSSE, cuando se agrupan dos clusters (cluster A y cluster B) formando un nuevo cluster R es:

$$\begin{aligned} \Delta SSE &= SSE(R) - \Big[SSE(A) + SSE(B)\Big] \\ &= \sum_{i \in R} \parallel x_i - \overline{x}(R) \parallel^2 \; - \sum_{i \in A} \parallel x_i - \overline{x}(A) \parallel^2 \; - \sum_{i \in B} \parallel x_i - \overline{x}(B) \parallel^2 \end{aligned} \tag{3.4}$$

donde $\overline{x}(R)$ es el centro de gravedad de R, $\overline{x}(A)$ y $\overline{x}(B)$ los centros de gravedad de A y B respectivamente y ΔSSE es el costo de fusionar el cluster A y B.

Al comienzo del algoritmo la SSE es cero, ya que cada elemento es un cluster. La SSE se va incrementando a medida que se van juntando los elementos y/o clusters en cada paso iterativo. El objetivo del método es monitorear que

la variación sea lo más pequeña posible.

En cada etapa se considera la SSE total, que es la suma de $SSE(C)$, es decir, la suma de los cuadrados de los errores de todos los clusters formados en dicha etapa de aglomeración, donde solo puede incrementar (o quedar igual) en cada nivel siguiente.

$$\begin{aligned}
\Delta SSE &= SSE(R) - SSE(A) - SSE(B) \\
&= \sum_{i \in R} \parallel x_i - \overline{x}(R) \parallel^2 - \sum_{i \in A} \parallel x_i - \overline{x}(A) \parallel^2 - \sum_{i \in B} \parallel x_i - \overline{x}(B) \parallel^2 \\
&= \sum_{i \in A} \parallel x_i - \overline{x}(R) \parallel^2 + \sum_{i \in B} \parallel x_i - \overline{x}(R) \parallel^2 - \sum_{i \in A} \parallel x_A - \overline{x}(A) \parallel^2 \\
&- \sum_{i \in B} \parallel x_i - \overline{x}(B) \parallel^2 \\
&= \sum_{i \in A} \Big[\parallel x_i - \overline{x}(R) \parallel^2 - \parallel x_i - \overline{x}(A) \parallel^2 \Big] + \sum_{i \in B} \Big[\parallel x_i - \overline{x}(R) \parallel^2 \\
&- \parallel x_i - \overline{x}(B) \parallel^2 \Big]
\end{aligned}$$

Operando, se llega a:

$$\Delta SSE = \frac{|A|\,|B|}{|R|}\ \|\overline{x}(A) - \overline{x}(B)\|^2 \tag{3.5}$$

El desarrollo completo de 3.5 se presenta en el Anexo B. Es así que el incremento de la suma de los errores al cuadrado es igual a la distancia euclídea entre los centros de gravedad de los cluster fusionados A y B, ponderada por los tamaños de dichos grupos.

Por otro lado, partiendo de la formulación de actualización propuesta por Kaufman [Kaufman, 2008], se obtiene que la disimilaridad entre dos clusters es la distancia ecuclídea respecto a sus centroides multiplicada por un factor:

$$d^2(A,B) = \frac{2\,|A|\,|B|}{|A|+|B|}\;\|\overline{x}(A)-\overline{x}(B)\|^2 \tag{3.6}$$

Bajo esta expresión es que se actualizan las distancias en cada paso iterativo de aglomeración, suponiendo que se fusionan los clusters A y B.

Se deduce entonces que el incremento de la suma de los errores al cuadrado es la mitad de la disimilaridad entre el cluster A y B:

$$\Delta SSE = \frac{1}{2}\,d^2(A,B)$$

Se verifica [Anexo: B] que la expresión de actualización de las distancias entre dos clusters que se fusionan, se puede expresar mediante la formulación de *Lance y Williams*, de la siguiente manera:

$$\begin{aligned} d^2(R,Q) = &\ \frac{2\,|R|\,|Q|}{|R|+|Q|}\;\|\overline{x}(R)-\overline{x}(Q)\|^2 \\ = &\ \frac{|A|+|Q|}{|R|+|Q|}\,d^2(A,Q) + \frac{|B|+|Q|}{|R|\;+\;|Q|}\,d^2(B,Q) \\ - &\ \frac{|Q|}{|R|+|Q|}\,d^2(A,B) \end{aligned}$$

Por lo tanto, la distancia entre los grupos fusionados queda determinada por:

$$\begin{aligned} d(R,Q) = &\ \sqrt{\frac{2\,|R|\,|Q|}{|R|+|Q|}\;\|\overline{x}(R)-\overline{x}(Q)\|^2} \\ = &\ \sqrt{\frac{|A|+|Q|}{|R|+|Q|}\,d^2(A,Q)\;+\;\frac{|B|+|Q|}{|R|+|Q|}\,d^2(B,Q)\;-\;\frac{|Q|}{|R|+|Q|}\,d^2(A,B)} \end{aligned}$$

3.5. Limitaciones y características

La naturaleza de los datos y la cantidad de observaciones son las principales restricciones al implementar un algoritmo de segmentación, pues se debe disponer de alguna matriz de distancia o disimilaridad. No es necesario contar

con los datos originales, pero es necesario disponer de una matriz de distancia.

La función de distancia que se debe emplear para cuantificar la similitud o proximidad entre los datos, es determinada por la naturaleza de éstos. Por lo tanto el problema a resolver surge del tamaño de la matriz de distancia, que es directamente proporcional a la cantidad de datos, siendo de $n(n-1)/2$ medidas entre las n observaciones totales.

La distancia para cada par de puntos se tiene que almacenar en una matriz, a medida que se va actualizando la misma en cada paso del proceso de aglomeración jerárquico. Por lo que una de las restricciones o limitaciones de la técnica es el costo computacional.

En el presente trabajo dadas las características de los datos tanto por la naturaleza como la cantidad de observaciones, se intenta simplificar el procedimiento utilizando las propiedades intrínsecas a los datos de estudio.

Cuando se dispone de datos cuantitativos, se suele utilizar una función de distancia como la euclídea para la construcción de la matriz de distancias, y se puede llegar a tener $n(n-1)/2$ distancias distintas.

Para el caso de tratarse de datos cualitativos, la cantidad de disimilaridades distintas también puede igualar a la cantidad de $n(n-1)/2$, pero a medida que aumenta el número de observaciones, es de esperar que se comiencen a presentar casos repetidos. La cantidad de variables es un factor que incrementa la posibilidad de que dos individuos sean distintos, pero dejando constante el número de variables, al incrementar el número de observaciones, éstas pueden comenzar a repetirse. Se pueden presentar casos donde se dispone de individuos cuyos atributos sean iguales; la medida de disimilaridad es cero.

Si bien esta situación puede darse tanto con datos cuantitativos como cualitativos, las características mencionadas se espera que sean más frecuentes al tratarse de datos cualitativos.

Entonces, lo que en principio puede considerarse una restricción para la implementación del algoritmo, puede ser útil si se observan las características mencionadas que hacen aparecer ceros en la matriz de disimilaridad.

3.6. Caso de Estudio: Distancia cero

En el presente trabajo se propone un método basado en el algoritmo aglomerativo jerárquico, según la técnica **Agnes** propuesta por Kaufman [Kaufman, 2008], utilizando como expresión de actualización de distancias la fórmula de *Lance y Williams* (ver fórmula 3.1).

El propósito es poder detectar los casos en que se encuentran disimilaridades iguales a cero y reducir la dimensión de la matriz de distancias, tomando ventaja de esta característica, con el fin de que la cantidad de observaciones no sea una limitación en la implementación del algoritmo, sino que sea un atributo aprovechable.

Sean n observaciones y m variables. Sea k la cantidad de observaciones únicas x_i con $i = 1 \dots k$ y $k \leq n$.

Se tiene entonces x_j tal que $j = \{k+1, \dots, n\}$ tal que $\exists\ x_i$ donde $x_i = x_j$ y $d(x_i, x_j) = 0$.

Mediante la implementación del algoritmo **Agnes**, se aglomeran aquellos elementos cuya distancia es la menor en la matriz de distancias, por lo que se juntan en primer instancia todas las observaciones que presentan distancia cero, o sea los clusters cuyos elementos son iguales.

La actualización de la matriz de distancias implica que la distancia relativa respecto a otro cluster Q sea calculada mediante la formulación 3.1.

Por lo tanto, cuando se agrupan dos elementos que son iguales cuya distancia es cero, al actualizar la matriz de distancias, dicha actualización depende solamente de α_i y α_j, pues los términos que contienen a β y a γ se anulan con los criterios *VML*, *VMC* y *Weigthed*; los coeficientes α_i y α_j se fijan en $1/2$. Para el criterio de *Average* y *Ward*, la actualización de las disimilaridades depende de la cardinalidad de los grupos que se estén fusionando.

	α_i	α_j
VML	$1/2$	$1/2$
VMC	$1/2$	$1/2$
$Weigthed$	$1/2$	$1/2$
$Average$	$\frac{\|A\|}{\|R\|}$	$\frac{\|B\|}{\|R\|}$
$Ward$	$\frac{\|A\|+\|Q\|}{\|R\|+\|Q\|}$	$\frac{\|B\|+\|Q\|}{\|R\|+\|Q\|}$

Tabla 3.2: Coeficientes de L-W según criterios, cuando se fusionan elementos cuya distancia es 0.

En cada paso de aglomeración con los criterios de *VML*, *VMC*, *Weigthed* y *Average*, la matriz de disimilaridad solo reduce su dimensión en una unidad por fila y por columna, permaneciendo invariantes las medidas de distancias del resto de la matriz.

Para el método de *Ward*, no solo se disminuye la cantidad de filas y columnas de la matriz de distancias, sino que se deben actualizar todos los pares de distancias que involucran a alguno de los clusters agrupados. Se actualizan dichas medidas de disimilaridad, según la cardinalidad de los clusters fusionados y la distancia que presentaban dichos clusters en un paso de aglomeración anterior, permaneciendo invariantes el resto de las distancias que no involucran a ninguno de los grupos unidos en el paso actual.

Entonces, es posible de esta manera ir reduciendo la matriz de distancias en cada paso de aglomeración, mientras existan clusters cuyas distancias sean cero.

Como resultado se obtiene una matriz de disimilaridad de dimensiones $k \times k$ a partir de la cual continúa el algoritmo de aglomeración.

A partir de esta instancia, la actualización de las disimilaridades depende también de la disimilaridad de los clusters que se fusionaron en pasos anteriores. Es decir, a partir del paso $n - k$, donde las distancias entre los elementos y/o clusters es distinta de cero.

3.7. Algoritmo Agnes Reducido (AGNES-R)

Se supone k observaciones únicas. Sin pérdida de generalidad asumimos que las observaciones únicas $\{1, \ldots, k\}$ se encuentran ordenadas, es decir que en la matriz de disimilaridad son las primeras k observaciones, seguidas por las observaciones repetidas.

Sea $a_i \in \{1, \ldots, k\}$ una observación única, y $a_j \in \{k+1, \ldots, n\}$ una observación repetida, tal que $d(a_i, a_j) = 0$, pues a_j es una observación repetida de a_i.

Es así que se elimina, de la matriz de disimilaridad la fila y la columna de la observación repetida y se fusionan las observaciones a_i y a_j, tal que $a_i \cup a_j = a_{i,j}$, formando un cluster de dos elementos. Se actualiza la disimilaridad según el criterio de clusterización seleccionado mediante la fórmula recursiva de *Lance y Williams*.

Se obtiene así, una matriz de $(n-1) \times (n-1)$ elementos con k (aún ordenados) grupos de observaciones únicas y $(n-1)-k$ elementos o grupos repetidos.

Nuevamente se identifica (mediante sorteo) otro par de observaciones o grupos cuya distancia sea cero y se repite el procedimiento, actualizando en cada paso la matriz de disimilaridad y guardando en un vector auxiliar la cantidad de observaciones que van conformando los $(n-k)$ grupos, hasta llegar a la etapa donde quedan exactamente k grupos únicos cuyas distancias entre clusters sean distintas a cero.

En este punto iterativo se tiene una matriz de disimilaridad reducida (Matriz Suficiente de Disimilaridad) y actualizada, que solo contiene clusters cuyas distancias no son cero y se dispone además de un vector $v : v = (v_1, \ldots, v_k)$, donde se va almacenando la cardinalidad de cada cluster formado en cada paso de aglomeración.

Observación: En la implementación en el caso que exista más de una observación con distancia minimal, se guarda con la posición de la observación que se presente en el último lugar encontrado. Los resultados obtenidos me-

diante sorteo y eligiendo la última posición encontrada son iguales. Se opta por realizar esta modificación para simplificar la programación.

3.8. Algoritmo AGNES-R Propuesto

Se propone partir de la matriz de distancia reducida (matriz de distancia construída a partir de *observaciones únicas*) para implementar el procedimiento de aglomeración, disminuyendo sustancialmente los cálculos en cada paso de fusión de grupos, pues se eliminan todas aquellas etapas donde existen observaciones que tienen distancia cero entre si.

Por lo que a menor cantidad de observaciones únicas se logra una mayor reducción de la matriz de disimilaridad de *input* del algoritmo, pasando de $n \times n$ a $k \times k$.

El vector de frecuencias v de observaciones repetidas se obtiene directamente de la matriz original de datos, o en su defecto a partir de la matriz de disimilaridad original. Se realizan los cálculos de actualización de disimilaridades a partir de la matriz reducida, utilizando las frecuencias absolutas de cada observación única (considerando el total de las observaciones).

Para poder utilizar la **Matriz Suficiente de Disimilaridad (MSD)**[1] (matriz reducida), es necesario además especificar el criterio de actualización de disimilaridades.

En el caso de *VMC*, *VML*, *Average* y *Weighted*, solo es necesario disponer del vector de frecuencias absolutas v de las observaciones únicas, pues hasta obtener la matriz reducida en el paso $n - k$, las disimilaridades permanecen invariantes, solo se reduce la dimensión de la matriz.

Para el caso de *Ward* la **MSD** con las observaciones únicas, es una matriz auxiliar que debe ser transformada, utilizando la fórmula propuesta por Kaufman [Kaufman, 2008]:

[1] Matriz construída a partir de *observaciones únicas*.

$$d^2(R,Q) = 2\,\frac{|A|\,|B|}{|R|}\,d^2(A,B)$$

Una vez realizada la reducción de la matriz de disimilaridad y luego transformada según el criterio de aglomeración (solo *Ward* requiere transformar), se procede a implementar el algoritmo de aglomeración jerárquico, utilizando para los posteriores cálculos el vector de frecuencias que refiere al paso $n-k$, el cual se va actualizando a medida que se comienzan a fusionar los sucesivos clusters dentro del algoritmo.

El procedimiento concluye al llegar a una matriz de disimilaridad de 2×2 donde en el siguiente paso ocurre la aglomeración de todas las observaciones originales, conformando un único cluster.

Como regla de determinación de cantidad de cluster seleccionados, es posible la utilización de los estadísticos R^2, $pseudo-T$ y $pseudo-F$ [1] a partir de la etapa donde se comienza con la fusión de grupos cuya distancia sea distinta de cero. Mientras que las distancias de los elementos y/o cluster sean cero, o se trate de clusters con un solo elemento no es posible cálcular dichos índices.

3.9. Ejemplo Algoritmo Propuesto

Se propone para ejemplificar cómo se obtienen iguales resultados a partir de la matriz de disimilaridad original, así como aplicando directamente la reducción de dimensiones y transformación de la misma (*Ward*).

Continuando con el ejemplo de la sección 2.7, que consta de 7 observaciones con las cuales se calcula la matriz de disimilaridad [Tabla: 3.3], se presenta la aglomeración mediante *Agnes* paso a paso hasta la etapa donde no hay más disimilaridades que sean cero entre clusters. Se aplican los 5 criterios de aglomeración planteados.

[1] Estos índices se podrán calcular dependiendo de la naturaleza de los datos.

	T_1	T_2	T_3	T_4	T_5	T_6	T_7
T_1	0	2	0	0	4	0	4
T_2	2	0	2	2	5	2	5
T_3	0	2	0	0	4	0	4
T_4	0	2	0	0	4	0	4
T_5	4	5	4	4	0	4	0
T_6	0	2	0	0	4	0	4
T_7	4	5	4	4	0	4	0

Tabla 3.3: Matriz de disimilaridades.

3.9.1. Método de *Ward*

Partiendo de la matriz de disimilaridad [Tabla: 3.3], se plantean los primeros pasos iterativos hasta la etapa donde los clusters aglomerados no presentan distancias que sean cero. Se debe recalcular la matriz de disimilaridad, utilizando la fórmula de *Lance y Williams* con el método de *Ward* en cada iteración. En el ejemplo, es necesario realizar 4 pasos recursivos para obtener como resultado la **Matriz Suficiente de Disimilaridad de Ward (MSDW)**.

$$d(T_i - T_k; T_j) = \left(\frac{n_i + n_j}{n_i + n_j + n_k}\right) d(T_i, T_j)^2 + \left(\frac{n_k + n_j}{n_i + n_j + n_k}\right) d(T_k, T_j)^2$$

Siendo:

T_i, T_j y T_k las observaciones i, j y k respectivamente.
n_i, n_j y n_k la cardinalidad de cada grupo.
$d(T_i, T_j)$, la disimilaridad entre T_i y T_j.

Cuando $J = 1$, se tiene:

$$d(T_5 - T_7; T_1) = \left(\frac{n_5 + n_1}{n_5 + n_1 + n_7}\right) d(T_5, T_1)^2 + \left(\frac{n_7 + n_1}{n_5 + n_1 + n_7}\right) d(T_7, T_1)^2$$

$$= \left(\frac{2}{3}4^2\right) + \left(\frac{2}{3}4^2\right)$$

$$= 4.6188$$

Cuando $J = 2$, se tiene:

$$d(T_5 - T_7; T_2) = \left(\frac{n_5 + n_2}{n_5 + n_2 + n_7}\right) d(T_5, T_2)^2 + \left(\frac{n_7 + n_2}{n_5 + n_2 + n_7}\right) d(T_7, T_2)^2$$

$$= \left(\frac{2}{3}5^2\right) + \left(\frac{2}{3}5^2\right)$$

$$= 5.7735$$

Cuando $J = 3$, se tiene:

$$d(T_5 - T_7; T_3) = \left(\frac{n_5 + n_3}{n_5 + n_3 + n_7}\right) d(T_5, T_3)^2 + \left(\frac{n_7 + n_3}{n_5 + n_3 + n_7}\right) d(T_7, T_3)^2$$

$$= \left(\frac{2}{3}5^2\right) + \left(\frac{2}{3}5^2\right)$$

$$= 4.6188$$

Cuando $J = 4$, se tiene:

$$d(T_5 - T_7; T_4) = \left(\frac{n_5 + n_4}{n_5 + n_4 + n_7}\right) d(T_5, T_4)^2 + \left(\frac{n_7 + n_4}{n_5 + n_4 + n_7}\right) d(T_7, T_4)^2$$

$$= \left(\frac{2}{3}4^2\right) + \left(\frac{2}{3}4^2\right)$$

$$= 4.6188$$

Cuando $J = 6$, se tiene:

$$d(T_5 - T_7; T_6) = \left(\frac{n_5 + n_6}{n_5 + n_6 + n_7}\right) d(T_5, T_6)^2 + \left(\frac{n_7 + n_6}{n_5 + n_6 + n_7}\right) d(T_7, T_6)^2$$

$$= \left(\frac{2}{3}4^2\right) + \left(\frac{2}{3}4^2\right)$$

$$= 4.6188$$

La nueva matriz de disimilaridad en el primer paso iterativo, aglomera los clusters 5 y 7, formando un nuevo cluster T_{57}:

	T_1	T_2	T_3	T_4	T_{57}	T_6
T_1	0.00	2.00	0.00	0.00	4.62	0.00
T_2	2.00	0.00	2.00	2.00	5.77	2.00
T_3	0.00	2.00	0.00	0.00	4.62	0.00
T_4	0.00	2.00	0.00	0.00	4.62	0.00
T_{57}	4.62	5.77	4.62	4.62	0.00	4.62
T_6	0.00	2.00	0.00	0.00	4.62	0.00

Tabla 3.4: Matriz de disimilaridad actualizada. Paso 1.

A partir de la nueva matriz de disimilaridad, se vuelve a repetir el procedimiento y se juntan los clusters 4 y 6. Los tamaños de los clusters en el paso 2 se almacenan en el vector de frecuencias $\mathbf{v} = \{1, 1, 1, 2, 2\}$.

	T_1	T_2	T_3	T_{46}	T_{57}
T_1	0.00	2.00	0.00	0.00	4.62
T_2	2.00	0.00	2.00	2.31	5.77
T_3	0.00	2.00	0.00	0.00	4.62
T_{46}	0.00	2.31	0.00	0.00	5.66
T_{57}	4.62	5.77	4.62	5.66	0.00

Tabla 3.5: Matriz de disimilaridad actualizada. Paso 2.

Se repite el procedimiento y en el paso 3, se fusionan el cluster $4-6$ con el cluster 3, obteniendo un vector de frecuencias actualizado $\mathbf{v} = \{1, 1, 3, 2\}$.

	T_1	T_2	T_{346}	T_{57}
T_1	0.00	2.00	0.00	4.62
T_2	2.00	0.00	2.45	5.77
T_{346}	0.00	2.45	0.00	6.20
T_{57}	4.62	5.77	6.20	0.00

Tabla 3.6: Matriz de disimilaridad actualizada. Paso 3.

Por último, se agrupan las observaciones $3-4-6$ con el grupo 1, se obtiene $\mathbf{v} = \{4, 1, 2\}$, donde la matriz actualizada en esta etapa no presenta disimilaridades que sean cero.

	T_{1346}	T_2	T_{57}
T_{1346}	0.00	2.53	6.53
T_2	2.53	0.00	5.77
T_{57}	6.53	5.77	0.00

Tabla 3.7: Matriz de disimilaridad actualizada. Paso 4.

3.9.2. Métodos Average, VMC, VML y Weighted

Partiendo de la matriz de disimilaridad [Tabla: 3.3], se presenta la actualización de la misma según los criterios de aglomeración *Average*, *Vecinos más cercano, Vecinos más lejano* y *Weighted*, a través de la fórmula de *Lance y William*, donde $d(T_i, T_k) = 0$ y $d(T_i, T_j) = d(T_k, T_j)$.

Método Average:

$$d(T_i - T_k; T_j) = \frac{n_i}{n_i + n_k}\, d(T_i, T_j) + \frac{n_k}{n_i + n_k}\, d(T_k, T_j)$$

$$= \left(\frac{n_i}{n_i + n_k} + \frac{n_k}{n_i + n_k}\right)\, d(T_i, T_j)$$

$$= \frac{n_i + n_k}{n_i + n_k}\, d(T_i, T_j) \;=\; d(T_i, T_j)$$

Método vecinos más cercanos:

$$d(T_i - T_k; T_j) = \frac{1}{2}\left[d(T_i, T_j) + d(T_k, T_j)\right] - \frac{1}{2}\left[d(T_i, T_j) - d(T_k, T_j)\right]$$

$$= \frac{1}{2}\left[d(T_i, T_j) + d(T_k, T_j)\right]$$

$$= \frac{1}{2}\left[2d(T_i, T_j)\right] = d(T_i, T_j)$$

Método vecinos más lejanos:

$$d(T_i - T_k; T_j) = \frac{1}{2}\left[d(T_i, T_j) + d(T_k, T_j)\right] + \frac{1}{2}\left[d(T_i, T_j) - d(T_k, T_j)\right]$$

$$= \frac{1}{2}\left[d(T_i, T_j) + d(T_k, T_j)\right]$$

$$= \frac{1}{2}\left[2d(T_i, T_j)\right] = d(T_i, T_j)$$

Método Weighted:

$$d(T_i - T_k; T_j) = \frac{1}{2}\left[d(T_i, T_j) + d(T_k, T_j)\right]$$

$$= \frac{1}{2}\left[2d(T_i, T_j)\right] = d(T_i, T_j)$$

A modo de ejemplo con los criterios de *Average, VMC, VML* y *Weighted*, la actualización de las disimilaridades mediante la fórmula de L-W, de los grupos 5 y 7 con el resto de los cluster se resume de la siguiente forma:

J			
1	$d(T_5 - T_7; T_1)$	$d(T_{5-7}, T_1)$	4
2	$d(T_5 - T_7; T_2)$	$d(T_{5-7}, T_2)$	5
3	$d(T_5 - T_7; T_3)$	$d(T_{5-7}, T_3)$	4
4	$d(T_5 - T_7; T_4)$	$d(T_{5-7}, T_4)$	4
6	$d(T_5 - T_7; T_6)$	$d(T_{5-7}, T_6)$	4

Tabla 3.8: Actualización de disimilaridad.

Cuando se fusiona el cluster 5 con el 7 se evalua la distancia del nuevo grupo fusionado $(5-7)$ respecto a los 5 grupos restantes, por ejemplo la disimilaridad $d(T_5 - T_7; T_1)$ es 4.

Una vez que se actualiza la disimilaridad de la unión de los clusters con el resto de los grupos, se eliminan las filas y las columnas involucradas en dicha unión, creando una nueva fila y columna que sustituye a las eliminadas. Las entradas de esta nueva columna y esta nueva fila son las disimilaridades calculadas en la actualización.

	T_1	T_2	T_3	T_4	T_{57}	T_6
T_1	0.00	2.00	0.00	0.00	4.00	0.00
T_2	2.00	0.00	2.00	2.00	5.00	2.00
T_3	0.00	2.00	0.00	0.00	4.00	0.00
T_4	0.00	2.00	0.00	0.00	4.00	0.00
T_{57}	4.00	5.00	4.00	4.00	0.00	4.00
T_6	0.00	2.00	0.00	0.00	4.00	0.00

Tabla 3.9: Matriz de disimilaridad actualizada. Paso 1.

	T_1	T_2	T_3	T_{46}	T_{57}
T_1	0.00	2.00	0.00	0.00	4.00
T_2	2.00	0.00	2.00	2.00	5.00
T_3	0.00	2.00	0.00	0.00	4.00
T_{46}	0.00	2.00	0.00	0.00	4.00
T_{57}	4.00	5.00	4.00	4.00	0.00

Tabla 3.10: Matriz de disimilaridad actualizada. Paso 2.

	T_1	T_2	T_{346}	T_{57}
T_1	0.00	2.00	0.00	4.00
T_2	2.00	0.00	2.00	5.00
T_{346}	0.00	2.00	0.00	4.00
T_{57}	4.00	5.00	4.00	0.00

Tabla 3.11: Matriz de disimilaridad actualizada. Paso 3.

	T_{1346}	T_2	T_{57}
T_{1346}	0.00	2.00	4.00
T_2	2.00	0.00	5.00
T_{57}	4.00	5.00	0.00

Tabla 3.12: Matriz de disimilaridad actualizada. Paso 4.

Es así, que para los cuatro métodos planteados, se llega a la **MSD** por **Agnes** (paso a paso) como por **AGNES-R** (eliminación de filas y columnas). A partir de la **MSD** se continúa el procedimiento aglomerativo con cualquiera de los dos algoritmos hasta la construcción total de la jerarquía de aglomeración.

Para la obtención de la **MSDW**, se utiliza como matriz auxiliar a **MSD**, transformando a la misma mediante la recursión propuesta por **AGNES-R**, donde se actualiza cada disimilaridad multiplicandola por un factor de la siguiente manera:

$$\sqrt{\frac{2\ n_i\ n_j}{n_i + n_j}\ d(T_i, T_j)^2}$$

La **MSDW** resulta en:

	T_1	T_2	T_5
T_1	0.00	2.53	6.53
T_2	2.53	0.00	5.77
T_5	6.53	5.77	0.00

Tabla 3.13: Matriz Suficiente de Disimilaridad de Ward.

Observación: para el caso de *Average, VMC, VML* y *Weighted* alcanza con la **MSD**, no es necesario transfromar la misma.

Capítulo 4

Aplicación

En el presente capítulo de aplicaciones prácticas de los algoritmos de **OMAR** y **AGNES-R**, se trabaja con un set de datos de trayectorias estudiantiles de la Universidad de la República, Facultad de Ciencias Económicas y de Administración.

En primer lugar se presenta un análisis descriptivo del set de datos, basado en la vida académica de los estudiantes entre el período 2012 - 2015. A continuación se aplica el algoritmo **OMAR** a las trayectorias (observaciones) únicas que presentan los datos. Se obtiene una matriz de disimilaridad, un vector de posiciones (en la matriz original de datos) de las observaciones únicas y un vector de frecuencias de la cantidad de veces que se repite cada observación única. Estos tres elementos, la matriz y los dos vectores, permiten la construcción de la matriz de disimilaridad general, o sea para todas las observaciones de la muestra.

En las siguientes secciones se presenta la implementación del algoritmo **AGNES-R** en el set de datos de las trayectorias estudiantiles. El punto de partida de dicha implementación podría ser considerar como insumo una matriz de disimilaridad reducida, con el vector de posiciones y el vector de frecuencias asociado; o se podría partir de la matriz general de disimilaridades con todas las observaciones[1].

[1]En caso de iniciar el algoritmo **AGNES-R** a partir de la matriz de disimilaridad reducida, se debe evaluar según el criterio de clusterización seleccionado si es necesario transformar o no a dicha matriz de distancia reducida. Se considera que las observaciones repetidas de las observaciones únicas, son elementos ya agrupados, pues la distancia entre las únicas y sus repeticiones, son cero. Esta característica se refleja en el vector de frecuencias que

En la aplicación presentada en este trabajo, se toma como punto de partida la segunda opción. El siguiente paso del algoritmo es buscar la **Matriz Suficiente de Disimilaridad**, construída solamente con las observaciones únicas según el criterio de segmentación seleccionado que en este caso es el método de *Ward*.

Se aplica la clusterización en base a la fórmula de actualización de *Lance y Williams* para el criterio de *Ward*, agrupando en primera instancia, todas las observaciones únicas con sus repetidas (presentan distancia igual a cero) y se actualiza en cada paso iterativo la matriz de disimilaridad.

Cuando no se detectan observaciones (o grupos) que presenten distancia cero en la matriz de distancia actualizada, se continúa el algoritmo realizando el mismo procedimiento, pero a parir de las observaciones que presenten mínima distancia entre sí. El algoritmo continúa actualizando mediante la fórmula recursiva de *Lance y Williams* y evaluando las distancias mínimas en cada iteración, hasta finalizar la jerarquía de agrupaciones.

Se debe evaluar cúal es el paso de iteración óptimo para determinar la cantidad de grupos resultantes. En este ejemplo de datos de trayectorias estudiantiles, se determina trabajar con 4 clusters con el objetivo funcional de desarrollar el procedimiento planteado, más que obtener una solución óptima.

4.1. Análisis Descriptivo

4.1.1. Fuentes de Información

Los datos de estudiantes de la Facultad de Ciencias Económicas y de Administración (FCEA), surgen de dos fuentes de información, registros del Sistema de Gestión de Bedelías (SGB) y el formulario de ingreso a FCEA de la División Estadística de la Dirección General de Planeamiento (DGPLAN).

Mediante el SGB se obtiene información acerca de la fecha de inscripción

indica con cuantas observaciones repetidas ha sido agrupada la observación única. A partir de esta instancia, el algoritmo continúa en la etapa donde no se detectan más disimilaridades iguales a cero en la matriz de distancias.

y la vida académica del estudiante a partir del ingreso a facultad, así como cursos, exámenes, aprobaciones, créditos, etc.
En el formulario de ingreso a facultad, se compila información variada del estudiante, como ser datos socio-demográficos, educación preuniversitaria, entre otros.

4.1.2. Datos analizados

Los datos analizados corresponden al período comprendido entre marzo 2012 - momento en que se implementa el nuevo Plan de Estudios - hasta diciembre 2015, suponiendo que la trayectoria de un estudiante en óptimas condiciones podría desarrollarse en 4 años curriculares.

Se excluye aquellos estudiantes con materias revalidadas al momento de la inscripción, es decir que se considera únicamente aquellos que acumulan 0 crédito a la fecha y sin actividad previa en FCEA. Las variables analizadas se presentan en tabla 4.1.

Se cuenta con 1761 observaciones (estudiantes), se trabaja con datos anonimizados.

Variable	Descripción
Xfem	Indicador de género; 1=mujer
edad_t	Intervalo de edad de ingreso a facultad
Xnac_lug	Lugar de nacimiento
	(Exterior - Interior - Montevideo)
Xsexto	Donde cursó 6to año de liceo
	(Interior Privada - Interior Pública -
	Montevideo Privada - Montevideo Pública)
Acum12, Acum13,	Créditos que tenía en cada uno de los períodos
Acum14, Acum15	
cat12_15	Trayectoria de estados desde marzo 2012
	hasta diciembre 2015
egreso	Indicador si egresó o no

Tabla 4.1: Descripción de variables.

Realizando una primera inspección se observa una cantidad significativa de

observaciones repetidas. Los datos son secuencias longitudinales, que presentan determinada periodicidad y con un alfabeto finito definido a partir de los créditos acumulados en cada año. Se distinguen los distintos estados a partir del alfabeto definido, tratandose de secuencias de trayectorias estudiantiles.

4.1.3. Trayectorias

Para la definición del alfabeto y de los estados, se realiza una simplificación de la evolución de una trayectoria estudiantil considerada como óptima, donde se establece que un alumno acumula a lo sumo 90 créditos por año. Se determina que aquellos estudiantes que no se inscribieron a ningún curso y/o examen, presentan un estado *inactivo* en el tiempo t. Los estudiantes que tuvieron actividad (inscripción a curso y/o exámen) pero no aprobaron en el período t, se consideran en el estado *SinCreditos*. Finalmente, se definen los estados *Primero*, *Segundo*, *Tercero*, *Cuarto* y *Quinto* según la acumulación de créditos de cada estudiante.

Estados	Etiqueta	Descripción
0	Sin Creditos	No ha generado créditos
1	Primero	Tiene entre 0 a 89 créditos acumulados
2	Segundo	Tiene entre 90 a 179 créditos acumulados
3	Tercero	Tiene entre 180 a 269 créditos acumulados
4	Cuarto	Tiene entre 270 a 359 créditos acumulados
5	Quinto	Tiene más de 359 créditos acumulados
*	Inactivo	Inactivo

Tabla 4.2: Estados posibles en la trayectoria de un estudiante.

4.1.4. Variables sociodemográficas

Las variables sociodemográficas refieren al sexo del estudiante, lugar de nacimiento, edad con la que ingresó a facultad e identicación de centro educativo en cuanto a calidad de privado o público, y ubicación del mismo (Exterior, interior o Montevideo).

Respecto a la distribución de ingreso de los estudiantes según el sexo y lugar de nacimiento, el 57, 6 % de ingresos a facultad en el año 2012 fueron mujeres. El 63, 4 % nacidos en Montevideo, 34, 6 % de estudiantes nacidos en el interior, y el restante 2 % estudiantes nacidos en el exterior.

Los estudiantes que cursaron sexto año de liceo en Montevideo, provienen en proporción similar de liceos públicos y privados, sin embargo, quienes cursan sexto año de liceo en el interior se distribuyen en un 90 % provenientes de liceos públicos y el 10 % de privados. Se observa que los ingresos procedentes de educación pública son mayoritariamente mujeres, mientras que de educación privada son similares en ambos sexos (ver figura 4.1).

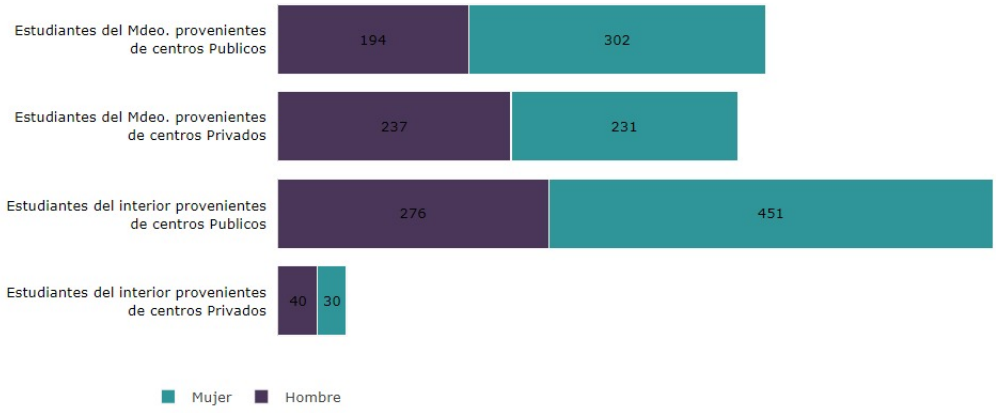

Figura 4.1: Sexo y lugar donde cursaron 6to año de liceo.

El 70 % de los ingresos a FCEA en el año 2012 fue de estudiantes que tenían entre 17 y 19 años, lo que significa que su ingreso fue inmediato a finalizar la educación media (ver figura 4.2).

El 8 % de los estudiantes que ingresaron en el año 2012, completaron sus estudios y egresaron, mientras que el restante 92 % no han culminado sus estudios en diciembre de 2015.

Las trayectorias que aparecen con mayor frecuencia son de estudiantes que en el primer año no computaron créditos y en años posteriores no registraron actividad dentro de FCEA. Con un 10.6 % de trayectorias, se presentan secuencias que se caracterizan por estudiantes que en los 4 períodos no superaron los 90 créditos. Solo el 4.1 % de los estudiantes transcurrieron de *Primero* a *Cuarto*, cambiando de estado en cada período considerado (ver tabla 4.3).

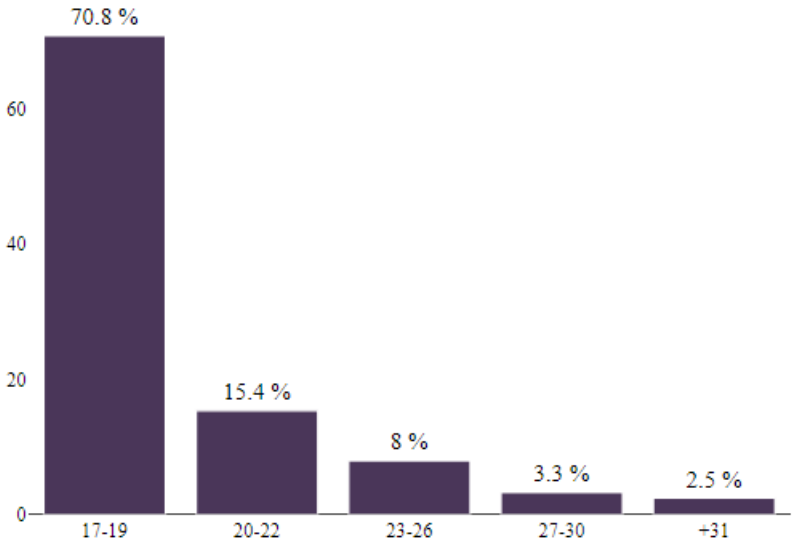

Figura 4.2: Ingresos según tramos de edad.

A2012	A2013	A2014	A2015	Frecuencia	Porcentaje
SinCreditos	Inactivo	Inactivo	Inactivo	209	11.9
Primero	Primero	Primero	Primero	187	10.6
Primero	Segundo	Segundo	Tercero	140	8.0
Primero	Segundo	Segundo	Segundo	107	6.1
Segundo	Segundo	Tercero	Cuarto	96	5.5
Primero	Primero	Segundo	Segundo	82	4.7
Primero	Inactivo	Inactivo	Inactivo	80	4.5
Primero	Primero	Inactivo	Inactivo	80	4.5
Primero	Segundo	Tercero	Tercero	74	4.2
Primero	Segundo	Tercero	Cuarto	72	4.1

Tabla 4.3: Matriz de *trayectorias únicas* (las 10 más freuentes).

El estado que se presenta con mayor frecuencia con un 34 % en las trayectorias de los estudiantes es *Primero*, que refiere a estudiantes que logran alcanzar al menos algún crédito pero no superan los 90. El estado con menor frecuencia es *Quinto* con una participación del 3 % (ver figura 4.3).

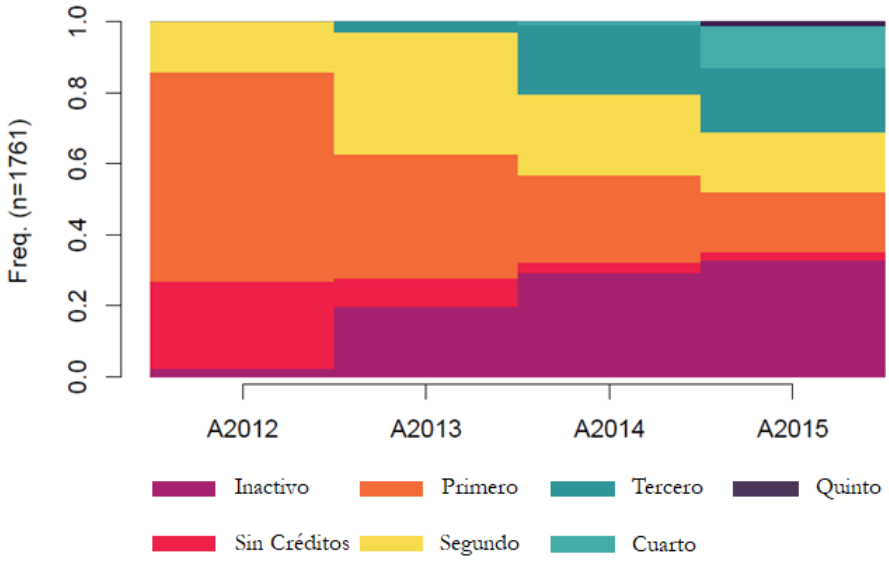

Figura 4.3: Frecuencia de secuencias de trayectorias educativas por período.

Se representa un diagrama de flujo (ver figura 4.4) con los 4 períodos considerados (t_i con $i = 1, ..., 4$; representados por los cuatro bloques verticales de color) y las distribuciones de los estados en cada uno de los períodos. El largo de cada barra de color significa el peso del estado en cada año (2012 al 2015) y el ancho de las uniones (barras grises) representa las frecuencias de transición entre un estado y otro de período a período. Por ejemplo, el estado que tiene mayor participación en 2012 es *Primero*. Los estudiantes que continúan en *Primero* en 2013 son los que presentan la mayor tasa de transición (de *Primero* a *Primero*, desde el período 1 al período 2), seguidos por los estudiantes que pasan de *Primero* a *Segundo* en 2013.

En el primer período t_i surgen 4 de los 7 estados presentes en las trayectorias estudiadas. Los estados se van ramificando llegando al año 2015, donde se observan los 7 estados presentes en las trayectorias educativas analizadas. Todos los estudiantes quedan representados según su estado en cada momento de la trayectoria estudiantil.

Dado el período t_i con $i = 1, ..., 4$, el estado *Tercero* ocurre por primera vez en t_2, el estado *Cuarto* aparece con muy baja participación en t_3, mientras que el estado *Quinto* aparece en el período t_4 también con un bajo peso. En este último año 2015 el estado que tiende a disminuir es el estado *SinCreditos*.

Es decir que los estudiantes alcanzan el último período generando al menos algún crédito, o terminan como estudiantes *Inactivos*, quienes aumentan en el transcurso del tiempo siendo los que mayor contribución tienen en el último año.

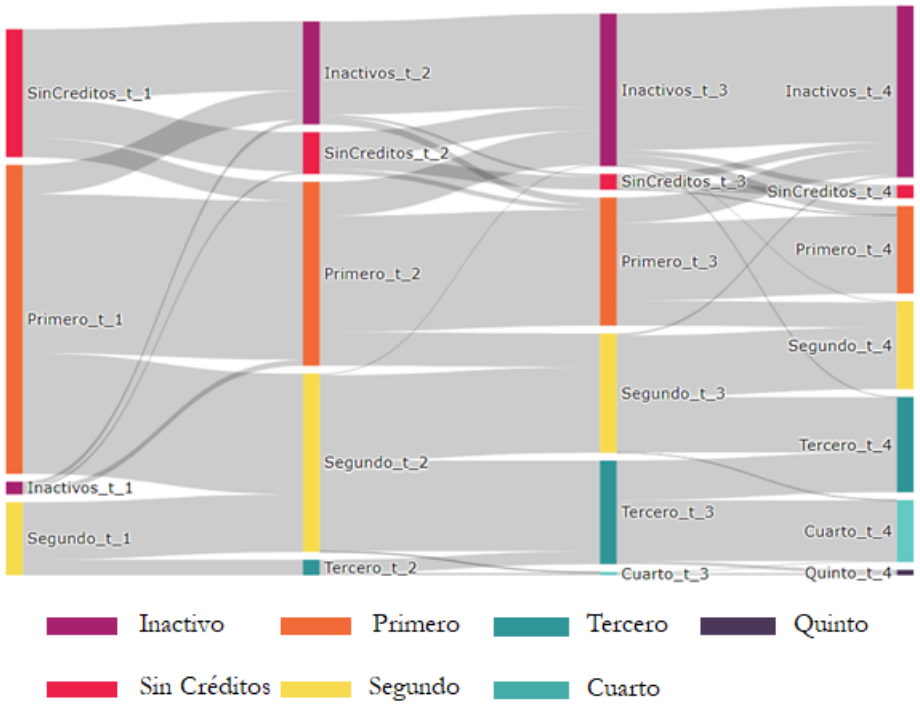

Figura 4.4: Diagrama Sankey de trayectorias educativas por período.

4.2. Aplicación OMAR

Como resultado de aplicar la técnica **OMAR**, se obtienen 70 *trayectorias únicas* de las 1761 observaciones. La trayectoria con mayor frecuencia es la de $SinCreditos|Inactivo|Inactivo|Inactivo$ con 209 repeticiones en total.

A partir de estas 70 *trayectorias únicas* se computa la matriz de disimilaridades y se almacena tanto el vector de frecuencias absolutas de cada trayectoria como el vector p de la primer posición en la que aparece la trayectoria única en el set de datos.

Por otra parte, a partir de la matriz de datos completa, se calculan las frecuencias de transición de un estado a otro, resultando en:

	Inactivo	SinCreditos	Primero	Segundo	Tercero	Cuarto	Quinto
Inactivo	0.8716	0.0409	0.0818	0.0044	0.0011	0	0
Sin Creditos	0.5440	0.3216	0.1344	0	0	0	0
Primero	0.1428	0	0.5680	0.2890	0	0	0
Segundo	0.0096	0	0	0.5524	0.4347	0.0032	0
Tercero	0	0	0	0	0.4438	0.5286	0.02743
Cuarto	0	0	0	0	0	0.3636	0.6363
Quinto	0	0	0	0	0	0	0

Tabla 4.4: Matriz de transición.

Cabe destacar que en esta aplicación, de datos provenientes de trayectorias educativas, no se cumple la propiedad de que la $p(a|b) = p(b|a)$, pues de hecho es imposible disminuir los créditos, es decir, pasar por ejemplo del estado *Tercero* al estado *Segundo*. Además es poco probable pasar con un rezago distinto a 1, de un estado que no sea ordinalmente consecutivo, como por ejemplo, pasar del estado *Primero* al *Cuarto*.

En la matriz de transición presentada en la tabla 4.4 se observa una mayor frecuencia relativa en permanecer de un período a otro en el estado *Inactivo*, siendo de 87 %, mientras que pasar del estado *Inactivo* al estado *Tercero* presenta la menor frecuencia relativa (1 %).

Una vez definida la matriz de transición se calcula la matriz de costos de sustitución (ver tabla 4.5), utilizando 2[1] como valor base al que se le sustraen las frecuencias de transición, tal que cada costo se calcula como:

$$w(a, b) = 2 - p(a|b) - p(b|a)$$

[1]Es el máximo costo que puede implicar pasar de un estado a otro si consideramos que tanto $p(a|b)$ y $p(b|a)$ pueden llegar a ser 0.

	Inactivo	Sin Creditos	Primero	Segundo	Tercero	Cuarto	Quinto
Inactivo	0	1.4150	1.7752	1.9859	1.9988	2	2
Sin Creditos	1.4150	0	1.8656	2	2	2	2
Primero	1.7752	1.8656	0	1.7109	2	2	2
Segundo	1.9859	2	1.7109	0	1.5652	1.9967	2
Tercero	1.9988	2	2	1.5652	0	1.4713	1.9725
Cuarto	2	2	2	1.9967	1.4713	0	1.3636
Quinto	2	2	2	2	1.9725	1.3636	0

Tabla 4.5: Matriz de costos de sustitución.

Por ejemplo (sin considerar los casos extremos donde el costo es 2 y 0), al alinear dos cadenas, se presenta un menor costo al pasar del estado *Cuarto* al estado *Quinto*, siendo el costo de 1.3636.

Finalmente se calcula la matriz de disimilaridad de *trayectorias únicas* aplicando el algoritmo de **OMAR**, con costos de sustitución basados en frecuencias de transición y costo *indel* igual a 1 a partir de las observaciones únicas. Con las disimilaridades calculadas, el vector de frecuencias y el vector de posiciones es posible generar la matriz de disimilaridades con el total de las observaciones.

4.3. Aplicación AGNES-R

Partiendo de la matriz de datos originales y con la matriz de disimilaridad generada tanto por la técnica de **OMA Clásico** de tamaño $n \times n$ como con la técnica **OMAR**, se aplica el algoritmo propuesto **AGNES-R** de forma de reducir la dimensión de la misma en una matriz auxiliar de dimensión $k \times k$. Es decir, se pasa de una matriz de disimilaridad de tamaño 1761×1761 a una matriz auxiliar de dimensión 70×70 que contiene solamente las disimilaridades de las observaciones únicas.

Como resultado de la implementación del algoritmo **AGNES-R**, se obtiene además el vector de frecuencias de cada una de las observaciones únicas y el vector auxiliar donde se guardan las posiciones de cada observación única.

Por otro lado, se debe seleccionar el criterio de aglomeración deseado, puesto que la reducción de la matriz de disimilaridad conlleva la transformación necesaria para la actualización de las disimilaridades de cada una de las etapas

de agrupamiento implícitas en dicha reducción.

Es así que se obtiene la **Matriz Suficiente de Disimilaridad (MSD)**[1], para la primer etapa de aglomeración donde no existen más observaciones cuyas distancias sean cero.

A continuación se presentan los resultados obtenidos a través del criterio de *Ward*. A fines ilustrativos se expone una solución basada en cuatro grupos, donde el *Cluster A* tiene 536 individuos (30.4 %), el *Cluster B* tiene 491 individuos (27.9 %), el *Cluster C* tiene 360 individuos (20.4 %) y el *Cluster D* tiene 374 individuos (21.2 %).

	Inactivo	SinCreditos	Primero	Segundo	Tercero	Cuarto	Quinto
Año 2012 Cluster A	19	357	160	0	0	0	0
Año 2013 Cluster A	320	124	92	0	0	0	0
Año 2014 Cluster A	483	53	0	0	0	0	0
Año 2015 Cluster A	484	44	8	0	0	0	0
Año 2012 Cluster B	11	13	434	33	0	0	0
Año 2013 Cluster B	3	0	195	293	0	0	0
Año 2014 Cluster B	5	0	84	402	0	0	0
Año 2015 Cluster B	9	0	0	294	187	1	0
Año 2012 Cluster C	0	0	147	213	0	0	0
Año 2013 Cluster C	0	0	0	308	52	0	0
Año 2014 Cluster C	0	0	0	0	349	11	0
Año 2015 Cluster C	0	0	0	0	134	208	18
Año 2012 Cluster D	13	61	300	0	0	0	0
Año 2013 Cluster D	24	17	333	0	0	0	0
Año 2014 Cluster D	26	0	348	0	0	0	0
Año 2015 Cluster D	85	0	287	2	0	0	0

Tabla 4.6: Distribución de estudiantes por cluster, por período y estado.

Para visualizar los patrones secuenciales y ayudar a la caracterización de cada cluster se presenta la distribución transversal (ver figura 4.5), o sea la cuantificación de los distintos estados en cada una de las posiciones (períodos), empleando los gráficos del paquete de R *TraMiner* [Gabadinho *et al.*, 2009] con la función *seqdplot()*.

[1]En el caso de seleccionar el criterio de *Ward*, se obtiene la **Matriz Suficiente de Disimilaridad de Ward**.

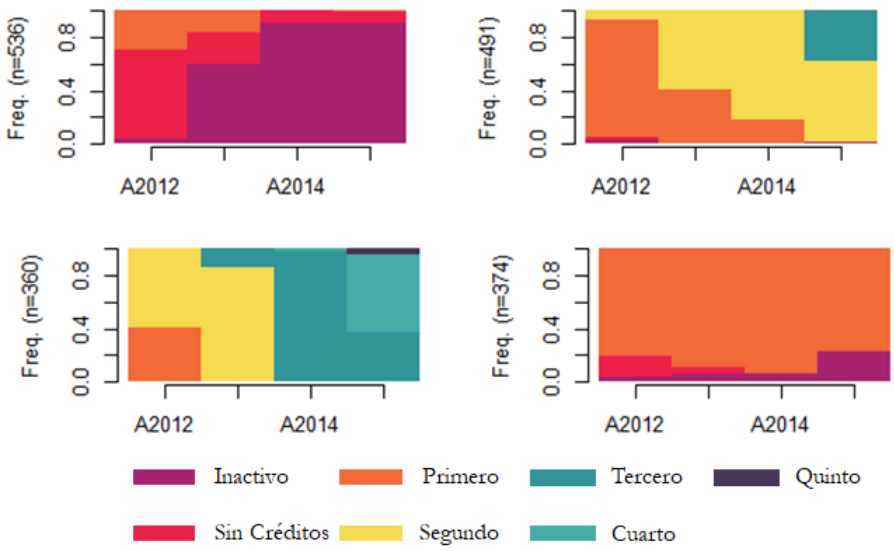

Figura 4.5: Distribución transversal de estados en los 4 períodos por Cluster.

Grupo A - Inactivos: Se caracteriza por tener individuos que no generaron créditos y estudiantes inactivos. El porcentaje de estudiantes inactivos aumenta con el tiempo, en 2012 la mayoría de los estudiantes se encuentran activos pero sin generar créditos. En el segundo año del período, el grupo de estudiantes que predomina son los inactivos, siendo estos el 60 %, alcanzando un 90 % de inactividad en los siguientes períodos.

Grupo B - Avance lento: Se caracteriza por estudiantes que incrementan sus créditos al pasar el tiempo, llegando a lo sumo a 270 créditos. El 88 % de los estudiantes finaliza el año 2012 habiendo generado a lo sumo 90 créditos. El 60 % de los individuos culmina los cuatro períodos con al menos 90 créditos y menos de 180 créditos, mientras que aproximadamente el restante 40 %, culmina el período de estudio en el estado *Tercero*. Es decir, se trata de estudiantes que avanzan en sus estudios por debajo del nivel óptimo esperado.

Grupo C- Avance bueno: Se caracteriza por individuos que realizan sus estudios de manera sostenida. No existen estudiantes sin actividad. Se aprecia un aumento en el nivel de créditos acumulados de forma progresiva, donde en el año 2013 aproximadamente el 80 % de los estudiantes se encuentra en el estado *Segundo*, avanzando en el siguiente año al estado *Tercero*. En el último período de estudio los estudiantes se distribuyen en un 37 % en el estado *Tercero* frente

a un 60 % que logran generar entre 270 y 360 créditos.

Grupo D- Eternos estudiantes: Se caracteriza por contener alumnos que se encuentran en el estado *Primero* durante todo el período de estudio. Es decir, estudiantes que han acumulado créditos sin llegar a tener más de 90.

La aplicación de la técnica **AGNES-R** presentada en este trabajo, no pretende encontrar una solución óptima de segmentación. El objetivo de dicha aplicación es meramente explicativo del algoritmo propuesto. Se realizaron implementaciones con **AGNES-R** utilizando los cinco criterios de clusterización, todos con la misma cantidad de grupos. Se encuentran disponibles los gráficos correspondientes a cada criterio en el [Anexo: C].

En las distintas implementaciones del procedimiento **AGNES-R** variando los criterios de clusterización se obtuvo idénticos resultados que al aplicar el método de **Agnes** propuesto por Kaufman [Kaufman, 2008] mediante el paquete de R *Cluster* [Maechler *et al.*, 2009] con la función *agnes()*.

Capítulo 5

Comentarios finales y trabajo a futuro

A modo de resumen y consideraciones finales, se destaca que el principal aporte de este trabajo es el desarrollo de dos métodos como alternativa a posibles dificultades computacionales que podrían presentarse tanto al implementar el procedimiento para generar disimilaridades (matriz de disimilaridades), así como el de clusterización aglomerativa jerárquica; que bajo determinadas características de los datos, se logra reducir la dimensión de las matrices de distancia necesarias como insumo para el cálculo de la segmentación.

El primer método propuesto que refiere a la generación de disimilaridades de datos secuenciales de tipo cualitativo, resulta como alternativa y complemento al procedimiento original Optimal Matching Analysis, permitiendo manipular grandes volúmenes de datos cuando estos presentan observaciones que se repiten a lo largo de la muestra.

La segunda propuesta complementa el modelado original propuesto por Kaufman [Kaufman, 2008] permitiendo trabajar y analizar un volumen de datos mayor, ahorrando costos y recursos computacionales.

Se implementan ambas propuestas en programas en base a lenguaje de programación R [Team, 2019], donde se realiza cada etapa de la clusterización bajo la técnica **Agnes** [Maechler *et al.*, 2009] y los diferentes criterios de agrupamiento que pueden ser empleados.

Se reproduce la técnica original **Agnes** verificando que se obtienen iguales resultados al implementar la técnica reducida **AGNES-R**. A su vez se reproduce también el método original **OMA Clásico** dividiendo y replicando cada una de sus etapas de programación y obteniendo idénticos resultados con la propuesta reducida **OMAR**.

Se evalúa la cantidad de cálculos computacionales que son requeridos en los métodos, donde se evidencia la reducción de operaciones, obteniendo iguales resultados.

Para la implementación de **OMA Clásico** son necesarios aproximadamente $\frac{n \times n}{2}$ cálculos (siendo n la cantidad de observaciones), mientras que para la propuesta **OMAR** se necesitan calcular aproximadamente $\frac{k \times k}{2}$ operaciones (siendo k la cantidad de observaciones únicas), por lo que a mayor diferencia entre n y k mayor es el ahorro computacional que se logra.

En cuanto a la implementación del algoritmo **Agnes**, la cantidad aproximada refiere al cálculo de tantas matrices de disimilaridad (menos uno) como observaciones disponibles, donde en cada paso iterativo ($n - 1$ etapas) se va disminuyendo en una unidad por fila y por columna la matriz de disimilaridad que requiere ser calculada, comenzando en la primer iteración con $\frac{n \times n}{2}$ cálculos, seguido por $\frac{n-1 \times n-1}{2}$ cálculos, hasta llegar a la última etapa donde se requiere realizar $\frac{2 \times 2}{2}$ cálculos. La propuesta de **AGNES-R** en tanto, puede reducir la cantidad de operaciones según la cantidad k de observaciones repetidas. Es decir, se requiere calcular $n - k - 1$ matrices de disimilaridad donde en el primer paso iterativo se comienza con una matriz de tamaño $\frac{n-k \times n-k}{2}$, disminuyendo en una unidad por fila y por columna en las sucesivas iteraciones hasta llegar a la última etapa requerida de $\frac{2 \times 2}{2}$ cálculos. Es así que a mayor diferencia entre la cantidad de observaciones n y la cantidad de observaciones únicas k, el ahorro de cálculos [1] y recursos computacionales es claramente mayor.

Los resultados obtenidos al aplicar **OMAR** y **AGNES-R** sobre datos de

[1] Considerar que en todos los casos se puede obviar el cálculo de la diagonal de cada matriz, pero con fines explicativos no se refleja el quitar de consideración el cálculo de la diagonal mencionada.

trayectorias estudiantiles de FCEA, son los mismos que los obtenidos al implementar **OMA Clásico** y **Agnes** variando los criterios de segmentación con las librerías disponibles de R [Team, 2019], *factorminer* [Kaufman, 2008] y *agnes* [Kaufman, 2008] respectivamente.

Por último, cabe destacar que en la implementación tanto de **AGNES-R** como en **OMAR**, el objetivo principal es bajar costos computacionales, de manera de lograr manipular efectivamente los datos e implementar dichas técnicas. Este objetivo se alcanza mediante la reducción de dimensiones que se logra en la matriz de disimilaridad, ya sea en su generación (**OMAR**) o como insumo (**AGNES-R**).

Como pasos a seguir a futuro, se propone en primer lugar, continuar trabajando en los programas desarrollados. En particular, se sugiere unificar los programas desarrollados para cada una de las técnicas. Se espera además, poder generar paquetes/librerías para disponibilizar en la comunidad estadística. Por último y de forma complementaria, se recomienda continuar trabajando en la mejora de la eficiencia computacional de los distintos programas desarrollados.

Referencias bibliográficas

Abbott, A. & Hrycak, A. (1990). Measuring resemblance in sequence data: An optimal matching analysis of musicians' careers. *American Journal of Sociology*, 96(1):144–185.

Gabadinho, A., Ritschard, G., Studer, M., & Müller, N. S. (2009). Mining sequence data in r with the traminer package: A users guide for version 1.2. *Geneva: University of Geneva*.

Hamming, R. W. (1950). Error detecting and error correcting codes. *The Bell System Technical Journal*, 29(2):147–160.

Kaufman, L. & Rousseeuw, P. J. (2008). *Finding groups in data: an introduction to cluster analysis*. John Wiley Sons, Ltd.

Kruskal, J. B. (1983). An overview of sequence comparison: Time warps, string edits, and macromolecules. *SIAM Review*, 25(2):201–237.

Lance, G. N. & Williams, W. T. (1966). A generalized sorting strategy for computer classifications. *Nature*, pp. 218–218.

Lesnard, L. (2006). Optimal matching and social sciences. Working Papers 2006-01, Center for Research in Economics and Statistics.

Levenshtein, V. I. (1966). Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707.

Maechler, M., Struyf, A., Hubert, M., & Hornick, K. (2009). Cluster: cluster analysis extended rousseeuw et al. *R library*.

Needleman, S. & Wunsch, C. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443—453.

Studer, M. & Ritschard, G. (2016). What matters in differences between life trajectories: A comparative review of sequence dissimilarity measures. *Journal of the Royal Statistical Society Series A (Statistics in Society)*, 179:481–511.

Team, R. C. (2019). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Videgain, K. (2015). Análisis longitudinal del registro nacional de alumnos sobre trayectorias educativas. *Recuperado de: http://publicaciones. inee. edu. mx/buscadorPub P, 3.*

Wikipedia contributors (2020). File: Needleman-wunsch pairwise sequence alignment.png — wikimedia commons, the free media repository. [Online; accessed 29-October-2020].

ANEXOS

Anexo A

Algoritmo Needleman-Wunsch

El algoritmo de Needleman - Wunsch (Needleman y Wunsch [1970]) fue propuesto por primera vez en 1970 por Saul Needleman y Christian Wunsch.

Fue una de las primeras aplicaiciones de programación dinámica para la comparación de secuencias biológicas, de dos proteínas (cadenas) considerando como unidad básica los aminoácidos (alfabeto) de cada proteína. La alineación óptima se define como el mayor número de aminoácidos de una proteína que pueden ser alineados con los de la otra proteína, permitiendo *gaps* (caracter nulo). Decir que una alineación es óptima implica que su *score* sea máximo. El algoritmo resuelve el problema de optimización almacenando los *scores* máximos de las soluciones de cada subproblema en lugar de recalcularlos.

Dadas dos secuencias a alinear a y b de tamaño m y n respectivamente, formadas por elementos de un alfabeto de símbolos, se representan las cadenas en una matriz S, matriz *Scoring* de 2 dimensiones donde están todos los posibles pares de combinaciones que se pueden considerar con los elementos de ambas cadenas.

La matriz Scoring $S_{(m+1\times n+1)}$, indica las similitudes entre los elementos de a y b, donde $S_{i,j}$ es la similitud que existe entre el elemento i de la secuencia a y el elemento j de la secuencia b.

Se define el parámetro d que indica cómo valorar que un símbolo no quede alineado con otro y que en su lugar se utilice un *gap* (caracter nulo).

El sistema de *Score* que se emplea en la matriz *Scoring* S, tiene tres posibilidades, *match* donde los caracteres son iguales, *mismatch* donde los caracteres son distintos e *indel* (Inserción y Eliminación) que sirve para alinear un caracter con un *gap*.

Los costos $C(a, b)$ asignados en el sistema de *score* entre las dos secuencias, utilizados por Needleman - Wunsch son:

match	+1
mismatch	- 1
indel	+1

Tabla A.1: Operadores.

En la primer fila de la matriz S se coloca una de las cadenas, a partir del elemento $S_{1,3}$ y en la primer columna de la matriz S se coloca la otra cadena a comparar, a partir del elemento $S_{3,1}$. Se fijan los elementos $S_{1,2} = S_{2,1} = gap$. Se fija el elemento $S_{2,2} = 0$. Los elementos de $S_{2,(3,\ldots n)}$ y $S_{(3,\ldots m),2}$ ambos se imputan aumentando en un *gap*.
El mejor *score* entre las dos cadenas será el valor de $S_{m,n}$. Cada valor de $S_{i,j}$ será la puntuación máxima entre el subproblema de $a(1 \ldots i)$ y $b(1 \ldots j)$, es decir que en cada posición de la matriz S, se tienen los máximos *scores* de los posibles alineamientos locales hasta la posición i, j.

$$S_{i,j} = \text{máx} \begin{cases} S_{i-1,j-1} + C_{i,j} \\ S_{i-1,j} + d \\ S_{i,j-1} + d \end{cases}$$

Siendo $C_{i,j}$ el costo de la transformación al alinear el elemento i de la cadena a y el elemento j de la cadena b.

Las celdas restantes se calculan y se guarda la información de la dirección (izquierda, diagonal o arriba) de donde se obtuvo el máximo *score* (se puede almacenar más de una dirección en cada cálculo, pues puede existir un empate al calcular el máximo).

Una vez completada la matriz S, se realiza el traceback del algoritmo desde la posición $S_{m,n}$ hasta $S_{1,1}$, con las direcciones guardadas en el cálculo de cada *score*.

El traceback queda determinado por el subconjunto de direcciones que se utilizaron para calcular el máximo de la celda inspeccionada, determinando la dirección traceback que une dicha celda con la celda que se utilizó para calcular el máximo *score*.

La última etapa del algoritmo es la de alinear las secuencias, haciendo una inspeccón de las direcciones del traceback. Si la dirección es horizontal o vertical, se inserta en la alineación un *gap*, en caso de que la dirección sea diagonal, se inserta el propio caracter correspondiente a la fila (columna) de la celda inspeccionada.

A modo de ejemplo se agrega la imagen (Fuente: [Wikipedia contributors, 2020]) de forma de explicar el traceback:

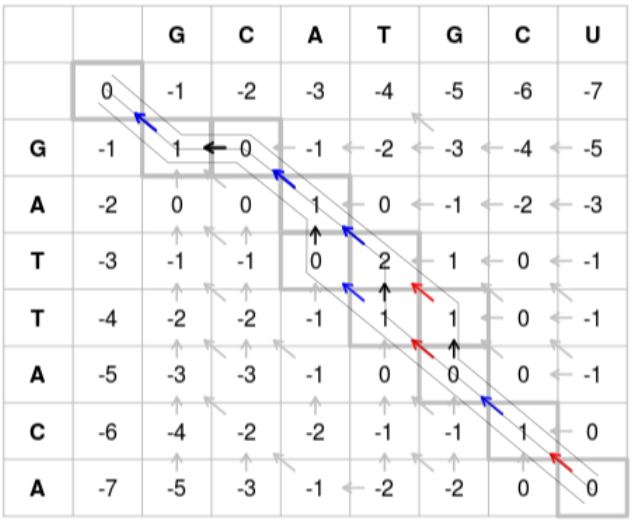

Figura A.1: Needleman-Wunsch: Alineación de pares de secuencias.

Anexo B

Deducción de la fórmula iterativa de Kaufman

Partiendo de la suma de los errores al cuadrado de un cluster C, como la suma de las distancias euclídias entre los elementos del cluster y su centroide $\overline{x}(C)$:

$$SSE = \sum_{i \in C} \parallel x_i - \overline{x}(C) \parallel^2$$

donde i es el i-ésimo elemento del cluster C.

Se define el incremento de la suma de los errores al cuadrado ΔSSE como el costo de fusionar dos cluster (cluster A y cluster B) formando un nuevo cluster R:

$$\begin{aligned} \Delta SSE &= SSE(R) - \Big[SSE(A) + SSE(B)\Big] \\ &= \sum_{i \in R} \parallel x_i - \overline{x}(R) \parallel^2 \sum_{i \in A} \parallel x_i - \overline{x}(A) \parallel^2 - \sum_{i \in B} \parallel x_i - \overline{x}(B) \parallel^2 \end{aligned}$$

donde $\overline{x}(R)$ es el centro de gravedad de R, $\overline{x}(A)$ y $\overline{x}(B)$ los centros de gravedad de A y B respectivamente.

Operando se tiene que:

$$\Delta SSE = SSE(R) - SSE(A) - SSE(B)$$

$$= \sum_{i \in R} \parallel x_i - \overline{x}(R) \parallel^2 - \sum_{i \in A} \parallel x_i - \overline{x}(A) \parallel^2 - \sum_{i \in B} \parallel x_i - \overline{x}(B) \parallel^2$$

$$= \sum_{i \in A} \parallel x_i - \overline{x}(R) \parallel^2 + \sum_{i \in B} \parallel x_i - \overline{x}(R) \parallel^2 - \sum_{i \in A} \parallel x_i - \overline{x}(A) \parallel^2$$

$$- \sum_{i \in B} \parallel x_i - \overline{x}(B) \parallel^2$$

$$= \sum_{i \in A} \Big[\parallel x_i - \overline{x}(R) \parallel^2 - \parallel x_i - \overline{x}(A) \parallel^2 \Big] + \sum_{i \in B} \Big[\parallel x_i - \overline{x}(R) \parallel^2$$

$$- \parallel x_i - \overline{x}(B) \parallel^2 \Big]$$

Obs 1: $\parallel a \parallel^2 - \parallel b \parallel^2 = (a+b)\,(a-b)$

Obs 2: $d^2(u,v) = \|u - v\|^2 = \|u\|^2 + \|v\|^2 - 2uv$

Obs 3: El centroide de R puede escibirse en función de A y B.

$$\overline{x}(R) = \frac{|A|}{|R|}\,\overline{x}(A) \; + \; \frac{|B|}{|R|}\,\overline{x}(B)$$

donde $|R| = |A| + |B|$

Entonces:

$$\begin{aligned}
\Delta SSE &= \sum_{i\in A}\left(\left[\Big(x_i-\overline{x}(R)\Big)+\Big(x_i-\overline{x}(A)\Big)\right]\left[\Big(x_i-\overline{x}(R)\Big)-\Big(x_i-\overline{x}(A)\Big)\right]\right)\\
&+\sum_{i\in B}\left(\left[\Big(x_i-\overline{x}(R)\Big)+\Big(x_i-\overline{x}(B)\Big)\right]\left[\Big(x_i-\overline{x}(R)\Big)-\Big(x_i-\overline{x}(B)\Big)\right]\right)\\
&=\sum_{i\in A}\left(\left[\Big(x_i-\overline{x}(R)\Big)+\Big(x_i-\overline{x}(A)\Big)\right]\left[\overline{x}(A)-\overline{x}(R)\right]\right)\\
&+\sum_{i\in B}\left(\left[\Big(x_i-\overline{x}(R)\Big)+\Big(x_i-\overline{x}(B)\Big)\right]\left[\overline{x}(B)-\overline{x}(R)\right]\right)\\
&=\left[\overline{x}(A)-\overline{x}(R)\right]\left[|A|\,\overline{x}(A)+|A|\,\overline{x}(A)-|A|\,\overline{x}(R)-|A|\,\overline{x}(A)\right]\\
&+\left[\overline{x}(B)-\overline{x}(R)\right]\left[|B|\,\overline{x}(B)+|B|\,\overline{x}(B)-|B|\,\overline{x}(R)-|B|\,\overline{x}(B)\right]\\
&=\left[\overline{x}(A)-\overline{x}(R)\right]|A|\left[\overline{x}(A)+\overline{x}(A)-\overline{x}(R)-\overline{x}(A)\right]\\
&+\left[\overline{x}(B)-\overline{x}(R)\right]|B|\left[\overline{x}(B)+\overline{x}(B)-\overline{x}(R)-\overline{x}(B)\right]\\
&=|A|\left[\overline{x}(A)^2-2\overline{x}(A)\,\overline{x}(R)-\overline{x}(R)^2\right]\\
&+|B|\left[\overline{x}(B)^2-2\overline{x}(B)\,\overline{x}(R)-\overline{x}(R)^2\right]
\end{aligned}$$

Como:

$$= |A| \ \| \overline{x}(A) - \overline{x}(R)\|^2 + |B| \ \| \overline{x}(B) - \overline{x}(R)\|^2$$

$$= |A| \ \left\| \overline{x}(A) - \left(\frac{|A|}{|R|} \overline{x}(A) + \frac{|B|}{|R|} \overline{x}(B) \right) \right\|^2$$

$$+ |B| \ \left\| \overline{x}(B) - \left(\frac{|A|}{|R|} \overline{x}(A) + \frac{|B|}{|R|} \overline{x}(B) \right) \right\|^2$$

$$= |A| \ \left\| \frac{|R| \ \overline{x}(A) - |A| \ \overline{x}(A) - |B| \ \overline{x}(B)}{|R|} \right\|^2$$

$$+ |B| \ \left\| \frac{|R| \ \overline{x}(A) - |A| \ \overline{x}(A) - |B| \ \overline{x}(B)}{|R|} \right\|^2$$

$$= |A| \ \left\| \frac{|B| \ \overline{x}(A) - |B| \ \overline{x}(B)}{|R|} \right\|^2 + |B| \ \left\| \frac{|A| \ \overline{x}(B) - |A| \ \overline{x}(A)}{|R|} \right\|^2$$

$$= \frac{|A| \ |B|^2}{|R|^2} \ \| \overline{x}(A) - \overline{x}(B)\|^2 \ + \ \frac{|A|^2 \ |B|}{|R|^2} \| \overline{x}(B) - \overline{x}(A)\|^2$$

$$= \frac{|A| \ |B|^2 + |A|^2 \ |B|}{|R|^2} \ \| \overline{x}(A) - \overline{x}(B)\|^2$$

$$= \frac{|A| \ |B| \ (|A| + |B|)}{|R|^2} \ \| \overline{x}(A) - \overline{x}(B)\|^2$$

$$= \frac{|A| \ |B|}{|R|} \ \| \overline{x}(A) - \overline{x}(B)\|^2$$

Reescribiendo:

$$\Delta SSE = \frac{|A| \ |B|}{|R|} \ \|\overline{x}(A) - \overline{x}(B)\|^2$$

Por otro lado, se plantea la propuesta de actualización de las distancias de Kaufman [Kaufman, 2008] al fusionar los clusters A y B formando un grupo R:

$$d^2(A,B) = \frac{2\,|A|\,|B|}{|A|+|B|}\;\|\overline{x}(A)-\overline{x}(B)\|^2$$

Donde se deduce que:

$$\Delta SSE = \frac{1}{2}\,d^2(A,B)$$

Es así que se puede plantear la fórmula de actualización de Lance y Williams del resto de las distancias al agrupar dos clusters con los restantes grupos existentes. Es decir, la distancia actualizada del grupo R formado por el cluster A y el cluster B, con el resto de los grupos Q:

$$\begin{aligned}
d^2(R,Q) &= \frac{2\,|R|\,|Q|}{|R|+|Q|}\;\|\,\overline{x}(R)-\overline{x}(Q)\,\|^2 \\
&= \frac{2\,|R|\,|Q|}{|R|+|Q|}\;\left\|\,\frac{|A|}{|R|}\,\overline{x}(A)\;+\;\frac{|B|}{|R|}\,\overline{x}(B)\;-\;\overline{x}(Q)\,\right\|^2 \\
&= \frac{2\,|R|\,|Q|}{|R|+|Q|}\;\left\|\,\frac{|A|}{|R|}\,\overline{x}(A)\;+\;\frac{|B|}{|R|}\,\overline{x}(B)\;-\;\frac{|A|+|B|}{|R|}\,\overline{x}(Q)\,\right\|^2 \\
&= \frac{2\,|R|\,|Q|}{|R|+|Q|}\;\left\|\,\frac{|A|}{|R|}\,\Big(\overline{x}(A)-\overline{x}(Q)\Big)\;+\;\frac{|B|}{|R|}\,\Big(\overline{x}(B)-\overline{x}(Q)\Big)\,\right\|^2
\end{aligned}$$

Por Obs 2:

$$\frac{2\,|R|\,|Q|}{|R|+|Q|}\left(\left\|\,\frac{|A|}{|R|}\,(\overline{x}(A)-\overline{x}(Q))\,\right\|^2\;+\;\left\|\,\frac{|B|}{|R|}\,(\overline{x}(B)-\overline{x}(Q))\,\right\|^2\right.$$

$$\left.-\,2\,\frac{|A|\,|B|}{|R|\,|R|}\,\Big(\overline{x}(A)-\overline{x}(Q)\Big)\,\Big(\overline{x}(B)-\overline{x}(Q)\Big)\right)$$

$$=\frac{2\,|R|\,|Q|}{|R|+|Q|}\left(\frac{|A|^2}{|R|^2}\;\|\overline{x}(A)-\overline{x}(Q)\|^2\;+\;\frac{|B|^2}{|R|^2}\;\|\,\overline{x}(B)-\overline{x}(Q)\|^2\right.$$

$$\left.-\,2\,\frac{|A|\,|B|}{|R|\,|R|}\,\Big(\overline{x}(A)-\overline{x}(Q)\Big)\,\Big(\overline{x}(B)-\overline{x}(Q)\Big)\right)$$

$$=\frac{2\,|Q|}{(|R|+|Q|)\,|R|}\left(\Big(|A|^2\;\|\overline{x}(A)-\overline{x}(Q)\|^2\;+\;|B|^2\;\|\overline{x}(B)-\overline{x}(Q)\|^2\,\Big)\right.$$

$$\left.-\;|A|\,|B|\,\Big(\,\|\overline{x}(A)-\overline{x}(Q)\|^2+\|\overline{x}(B)-\overline{x}(Q)\|^2-\|\overline{x}(A)-\overline{x}(B)\|^2\,\Big)\right)$$

$$=\frac{2\,|Q|}{(|R|+|Q|)\,|R|}\left(\Big(|A|^2\;+\;|A|\,|B|\Big)\;\|\overline{x}(A)-\overline{x}(Q)\|^2\right.$$

$$\left.+\;\Big(|B|^2\;+\;|A|\,|B|\Big)\,\|\overline{x}(B)-\overline{x}(Q)\|^2\;-\;|A|\,|B|\,\|\overline{x}(A)-\overline{x}(B)\|^2\right)$$

$$=\frac{2\,|Q|}{|R|+|Q|}\,\Big(|A|\;\|\overline{x}(A)-\overline{x}(Q)\|^2\;+\;|B|\;\|\overline{x}(B)-\overline{x}(Q)\|^2\,\Big)$$

$$-\;\frac{|A|\,|B|}{|R|}\,\|\overline{x}(A)-\overline{x}(B)\|^2$$

Lo que verifica la expresión de Lance y Williams.

$$d^2(R,Q) = \frac{2\,|A|\,|Q|}{|R|+|Q|}\,\frac{|A|+|Q|}{2\,|A|\,|Q|}\,d^2(A,Q) \;+\; \frac{2\,|B|\,|Q|}{|R|+|Q|}\,\frac{|B|+|Q|}{2\,|B|\,|Q|}\,d^2(B,Q)$$

$$-\;\frac{2\,|Q|}{|R|+|Q|}\,\frac{|A|\,|B|}{|R|}\,\frac{|A|+|B|}{2\,|A|\,|B|}\,d^2(A,B)$$

$$=\;\frac{|A|+|Q|}{|R|+|Q|}\,d^2(A,Q) \;+\; \frac{|B|+|Q|}{|R|+|Q|}\,d^2(B,Q)$$

$$-\;\frac{|Q|}{|R|+|Q|}\,d^2(A,B)$$

Anexo C

Aplicación OMAR y AGNES-R

Se presentan resultados de la aplicación de la técnica **OMAR** y **AGNES-R** sobre los datos de trayectorias educativas de estudiantes de FCEA. Se realizaron implementaciones de la segmentación utilizando los cinco criterios de clusterización, todos con la misma cantidad de grupos pre establecidos (k=4). Previamente se generaron las disimilaridades entre observaciones a partir de la aplicación del algoritmo **OMAR**, que fueron utilizadas como insumo para desarrollar la agrupación mediante **AGNES-R**.

En el presente trabajo no se realiza discusión ni validación de resultados de segmentación, así como no se evalúa la cantidad de clusters óptimos establecidos. Tampoco se discute o evalúa los criterios utilizados para definir disimilaridades entre observaciones. El objetivo fundamental de este trabajo es la implementación tanto de **OMAR** como **AGNES-R** al aplicar las diferentes técnicas propuestas reducidas en sustitución de los algoritmos originales. Por lo tanto, los resultados presentados son de carácter ilustrativo.

Se logró reproducir el algoritmo de clusterización original *Agnes* propuesto por Kaufman así como también el algoritmo **OMA Clásico**, extendiendo las soluciones a conjuntos de datos grandes, con variables de tipo cualitativas, posibilitando la reproducción de ambos algoritmos, eludiendo y solventando la limitación computacional que implican las versiones originales.

Para visualizar los patrones secuenciales de cada cluster mediante la distribución transversal de los estados, se presentan los gráficos empleando el

paquete del software libre R-Project [Team, 2019] *TraMiner* [Gabadinho *et al.*, 2009] con la función *seqdplot()*.

Criterio: Vecinos más cercanos - Simple linkage

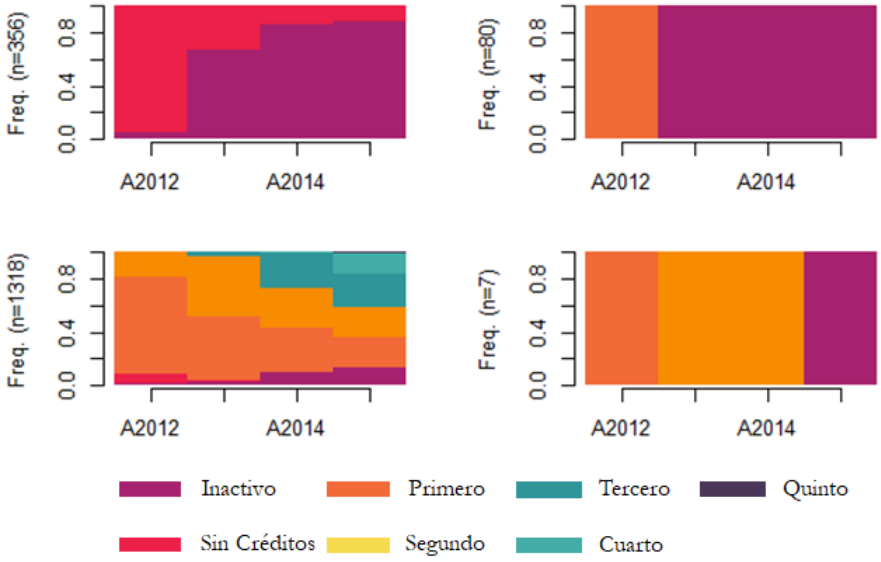

Figura C.1: Distribución transversal de estados: VMC.

Criterio: Vecinos más lejanos - Complete linkage

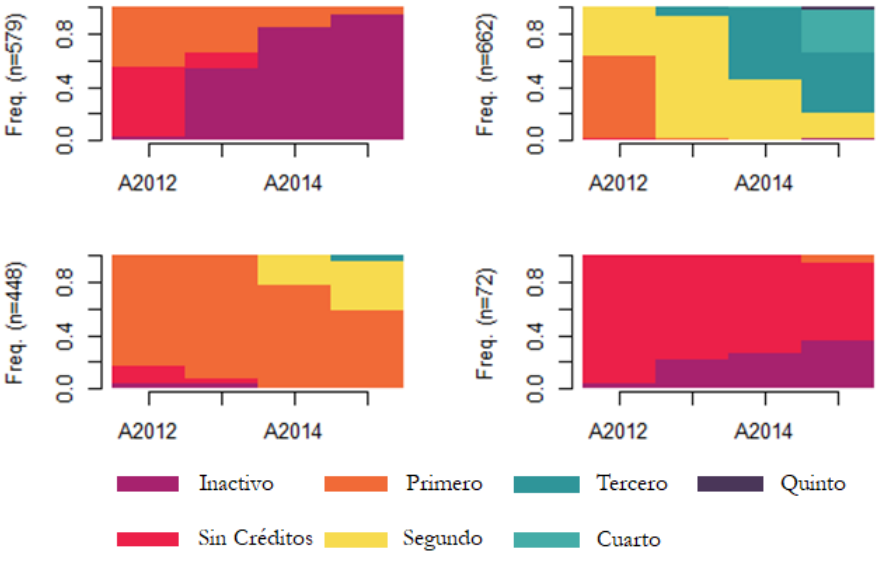

Figura C.2: Distribución transversal de estados: VML.

Criterio: Average - Unweighted pair group average method

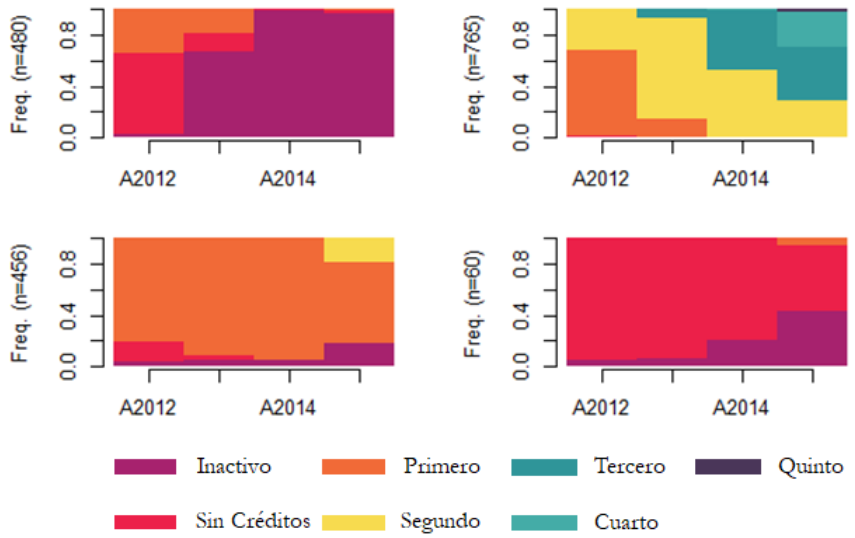

Figura C.3: Distribución transversal de estados: Average.

Criterio: Weighted Average Linkage

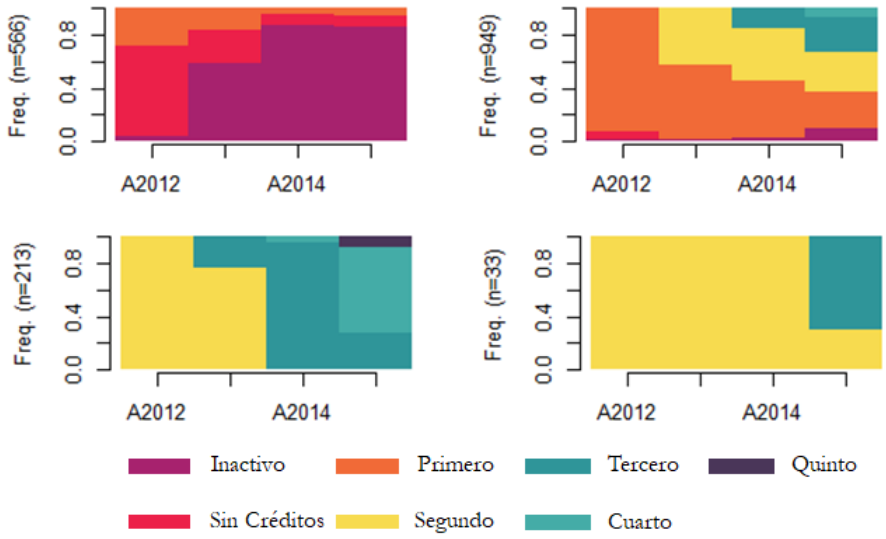

Figura C.4: Distribución transversal de estados: Weighted.

Anexo D

Programas desarrollados

En este capítulo se presentan y describen los programas desarrollados para los distintos algoritmos de **OMAR**, **AGNES-R** y para aplicar reglas de detención. Todos los algoritmos desarrollados se pueden ejecutar por etapas siguiendo en orden los scripts asociados. Para los algoritmos que refieren a **AGNES-R**, también se pueden ejecutar directamente en un único programa que consolida todas las etapas (scripts) necesarias. En la tabla D.1 se presenta un resumen de los programas utilizados en cada método.

Para implementar **OMAR**, se deben ejecutar en orden los primeros 5 programas en forma consecutiva.

Para la técnica **AGNES-R**, si bien se puede implementar paso a paso, ejecutando en forma consecutiva los scripts indicados (siguiendo el orden) según el criterio que se desee implementar, también se puede ejecutar directamente mediante el script 12, donde se debe indicar la matriz de datos, la matriz de disimilaridad y el criterio de segmentación que se desee emplear (*ward, average, weighted, vmc -Vecino más cercano- y vml -Vecino más lejano-*). Para implementar **AGNES-R** paso a paso con criterio *ward* por ejemplo, se deben ejecutar los scripts 1, 6, 7, 8, 9 y 12, o se puede ejecutar directamente mediante el script 12 que llama a los scripts anteriores.

Los scripts de reglas de detención de las segmentaciones realizadas se pueden ejecutar en etapas o directamente mediante el script 11 que llama a los programas anteriores necesarios. El programa de reglas de detención genera

la segmentación según el criterio definido previamente y brinda información sobre el cálculo de *pseudoF*, *pseudoT* y R^2 para cada iteración que se genera en la segmentación implementada.

Algoritmo	OMA	Cluster Ward	Cluster (VML-MLC-A-W)	Reglas de detención
`1 - secuencias`	X	X	X	X
`2 - Transicion`	X			
`3 - SustitutionCost`	X			
`4 - OMA`	X			
`5 - Disimilaridad`	X			
`6 - AuxYn`		X	X	X
`7 - buscoMasChico`		X	X	X
`8 - TransformerWard`		X		X
`9 - MidoDistancias`		X	X	X
`10 - MidoDistanciasyvalido`				
`11 - LaFuncionValido`				**X**
`12 - Cluster`		**X**	**X**	
`13 - TransformerComun`			X	

Tabla D.1: Programas utilizados en cada método.

Algorithm 1 `secuencias`

Genera una matriz con secuencias de estados a partir de una matriz de datos y un vector de estados.

Entrada:

datos: matriz de datos secuenciales de $n \times m$, donde n es la cantidad de observaciones y m la cantidad de períodos considerados.

vectorEstados: vector de estados que contiene las etiquetas de los estados, de igual tamaño que el alfabeto a utilizar, los estados deben estar ordenados según un orden alfa numérico determinado a priori.

Salida :

Secuencias: matriz de secuencias de estados de tamaño $n \times m$, donde n es la cantidad de observaciones y m la cantidad de períodos considerados.

Algorithm 2 `Transicion`

Calcula la frecuencia relativa de pasar de un estado a otro. Dichas frecuencias son consideradas como la probabilidad de $P(A|B)$ y $P(B|A)$ con un rezago de 1 (lag=1), siendo A y B dos estados distintos.

Entrada:

Secuencias: matriz de secuencias de estados de tamaño $n \times m$, donde n es la cantidad de observaciones y m la cantidad de períodos considerados.

vectorEstados: vector de estados que contiene las etiquetas de los estados, de igual tamaño que el alfabeto a utilizar, los estados deben estar ordenados según un orden alfa numérico determinado a priori.

Salida :

matrizTrans: matriz con las tasas de transición, de tamaño $k \times k$, donde k es la cantidad de estados.

Algorithm 3 `SustitutionCost`

Calcula los costos de sustitución de alinear estado a estado en el tiempo t de una secuencia con otra, calculando cada costo como $2 - P(A|B) - P(B|A)$. Los costos de sustitución se calculan a partir de la matriz de transición.

Entrada:

vectorEstados: vector de estados que contiene las etiquetas de los estados, de igual tamaño que el alfabeto a utilizar, los estados deben estar ordenados según un orden alfa numérico determinado a priori.

matrizTrans: matriz con las tasas de transicion, de tamaño $k \times k$, donde k es la cantidad de estados.

Salida :

MatrizSC: matriz con los costos de sustitución, de tamaño $k \times k$, donde k es la cantidad de estados.

Algorithm 4 `OMA`

Calcula la disimilaridad entre dos secuencias a partir de la matriz de costos de sustitución, con indel=1.

Entrada:

Sec1: vector que contiene una secuencia.

Sec2: vector que contiene una secuencia.

MatrizSC: matriz con los costos de sustitución, de tamaño $k \times k$, donde k es la cantidad de estados.

Indel=1, por defecto es 1.

Salida :

Valor de disimilaridad entre la *Sec1* y *Sec2*.

Algorithm 5 `Disimilaridad`

Construye la matriz de disimilaridades entre secuencias, a partir de las disimilaridades calculadas con la función OMA.

Entrada:

Secuencias: matriz de secuencias de estados de tamaño $k \times m$, donde k es la cantidad de observaciones únicas y m la cantidad de períodos considerados.
vectorEstados: vector de estados que contiene las etiquetas de los estados, de igual tamaño que el alfabeto a utilizar, los estados deben estar ordenados según un orden alfa numérico determinado a priori.
MatrizSC: matriz con los costos de sustitución, de tamaño $k \times k$, donde k es la cantidad de estados.
matrizTrans: matriz con las tasas de transición, de tamaño $k \times k$, donde k es la cantidad de estados.

Salida :

matrizDiss: matriz de disimilaridades de tamaño $u \times u$, donde u es la cantidad de observaciones únicas.

Algorithm 6 `AuxYn`

Genera un vector de frecuencias absolutas de cada secuencia única y una lista con las posiciones de las secuencias repetidas.

Entrada:

datos: matriz de datos secuenciales de $n \times m$, donde n es la cantidad de observaciones y m la cantidad de períodos considerados.
Secuencias: matriz de secuencias de estados de tamaño $n \times m$, donde n es la cantidad de observaciones y m la cantidad de períodos considerados.
matrizDiss: matriz de disimilaridades de tamaño $u \times u$, donde u es la cantidad de observaciones únicas.

Salida :

v: vector que contiene las cantidades de secuencias repetidas existentes, manteniendo el orden original de los datos.
aux: lista que contiene en cada elemento, la ubicación de las secuencias repetidas, manteniendo el orden de los datos originales.

Algorithm 7 `buscoMasChico`

Busca el valor mínimo, en la triangular superior (sin incluir la diagonal) de la matriz de disimilaridad. En caso de que el mínimo no sea único, se queda con la última posición encontrada.

Entrada:

matrizDiss: matriz de disimilaridades de tamaño $u \times u$, donde u es la cantidad de observaciones únicas.

valorChico: valor más chico que hay en la matriz triangular superior (sin incluir la diagonal) de la matriz de disimilaridades.

Salida :

sorteado: vector de tamaño 2, el cual contiene la ubicación (fila x columna) de este valor más pequeño.

Algorithm 8 `TransformerWard`

Transforma la matriz de disimilaridad de secuencias únicas, para utilizar el método de Ward.

Entrada:

datos: matriz de datos secuenciales de $n \times m$, donde n es la cantidad de observaciones y m la cantidad de períodos considerados.

Secuencias: matriz de secuencias de estados de tamaño $n \times m$, donde n es la cantidad de observaciones y m la cantidad de períodos considerados.

matrizDiss: matriz de disimilaridades de tamaño $u \times u$, donde u es la cantidad de observaciones únicas.

Salida :

matrizDissWard: matriz de disimilaridades Ward de tamaño $u \times u$, donde u es la cantidad de observaciones únicas.

v: vector que contiene las cantidades de secuencias repetidas existentes, manteniendo el orden original de los datos.

aux: lista que contiene en cada elemento, la posición de las secuencias repetidas de la matriz de datos.

Algorithm 9 `MidoDistancias`

Existen cinco funciones distintas como criterio de aglomeración: Average, Vecinos más cercanos, Vecinos más lejanos, Ward y Weighted. Cada una de ellas permite actualizar la matriz de disimilaridad a través de la fórmula de Lance y Williams.

Entrada:

datos: matriz de datos secuenciales de $n \times m$, donde n es la cantidad de observaciones y m la cantidad de períodos considerados.

matrizDiss: matriz de disimilaridades de tamaño $u \times u$, donde u es la cantidad de observaciones únicas.

sorteado: vector que contiene la posición del elemento más chico que hay en la matriz de disimilaresdes.

v: vector que contiene las cantidades de secuencias repetidas existentes, manteniendo el orden original de los datos.

aux: lista que contiene en cada elemento la ubicación de las secuencias repetidas, manteniendo el orden de los datos originales.

Salida :

matriz: matriz de disimilaridades actualizada mediante Lance y Williams, correspondiente al paso de aglomeración, reduciendo en una dimensión tanto filas como columnas.

v: vector que contiene las cantidades de secuencias repetidas existentes, manteniendo el orden original de los datos.

aux: lista que contiene en cada elemento, la posición de las secuencias repetidas de la matriz de datos.

Algorithm 10 `MidoDistanciasyvalido`

Existen cinco funciones distintas como criterio de aglomeración: Average, Vecinos más cercanos, Vecinos más lejanos, Ward y Weighted. Cada una de ellas permite actualizar la matriz de disimilaridad a través de la fórmula de Lance y Williams. En cada paso iterativo calcula el pseudoF, pseudoT y el coeficiente de determinación R^2.

Entrada:

datos: matriz de datos secuenciales de $n \times m$, donde n es la cantidad de observaciones y m la cantidad de períodos considerados.

matrizDiss: matriz de disimilaridades de tamaño $u \times u$, donde u es la cantidad de observaciones únicas.

sorteado: vector que contiene la posición del elemento más chico que hay en la matriz de disimilaresdes.

v: vector que contiene las cantidades de secuencias repetidas existentes, manteniendo el orden original de los datos.

aux: lista que contiene en cada elemento, la posición de las secuencias repetidas de la matriz de datos.

Salida :

cluster: lista que contiene la matriz de disimilaridad, el vector de frecuencias v y una lista *aux* de las posiciones de las observaciones pertenecientes a cada cluster existente en cada paso de aglomeración.

Algorithm 11 `LaFuncionValido`

Entrada:

datos: matriz de datos secuenciales de $n \times m$, donde n es la cantidad de observaciones y m la cantidad de períodos considerados.

matrizDiss: matriz de disimilaridades de tamaño $u \times u$, donde u es la cantidad de observaciones únicas.

metodo: se puede optar por los siguientes criterios: ward, average, weighted, vmc (Vecino más cercano) y vml (Vecino más lejano).

Salida :

salida: matriz de tamaño $3 \times n$, que contiene el resultado de R^2, *pseudoF* y *pseudoT*, para cada paso de unión de los cluster.

Algorithm 12 `Cluster`

Entrada:

datos: matriz de datos secuenciales de $n \times m$, donde n es la cantidad de observaciones y m la cantidad de períodos considerados.

metodo: se puede optar por los siguientes criterios: ward, average, weighted, vmc (Vecino más cercano) y vml (Vecino más lejano).

matrizDiss: matriz de disimilaridades de tamaño $u \times u$, donde u es la cantidad de observaciones únicas.

Salida :

cluster: lista que contiene la matriz de disimilaridad, el vector de frecuencias v y una lista aux de las posiciones de las observaciones pertenecientes a cada cluster existente en cada paso de aglomeración.

Algorithm 13 `TransformerComun`

Entrada:

datos: matriz de datos secuenciales de $n \times m$, donde n es la cantidad de observaciones y m la cantidad de períodos considerados.

Secuencias: matriz de secuencias de estados de tamaño $n \times m$, donde n es la cantidad de observaciones y m la cantidad de períodos considerados.

matrizDiss: matriz de disimilaridades de tamaño $u \times u$, donde u es la cantidad de observaciones únicas.

Salida :

lista: lista que contiene la matriz de disimilaridad, el vector de frecuencias v y una lista aux de las posiciones de las observaciones pertenecientes a cada cluster existente en cada paso de aglomeración. Además contiene una matriz con trayectorias únicas $trayUnicasDatos$.
