Sección de Matemáticas
Universidad de La Laguna

Belinda Hazel Cáceres Barrera

# *Homología persistente aplicada al estudio genético de poblaciones*

## Persistent homology applied to genetic study of populations

Trabajo Fin de Grado
Grado en Matemáticas
La Laguna, Junio de 2023

Dirigido por
*Carlos González Alcón*
*Josué Remedios Gómez*

*Carlos González Alcón*
*Matemáticas, Estadística e Investigación Operativa*
*Universidad de La Laguna*
*38200 La Laguna, Tenerife*

*Josué Remedios Gómez*
*Matemáticas, Estadística e Investigación Operativa*
*Universidad de La Laguna*
*38200 La Laguna, Tenerife*

# Resumen · Abstract


### *Resumen*

*El Análisis Topológico de Datos es una disciplina que combina conceptos y técnicas de la topología con el análisis de datos para estudiar y comprender la estructura subyacente en conjuntos de datos complejos. Su objetivo es revelar patrones y características importantes que pueden estar ocultos en los datos y que no son fácilmente detectables mediante métodos tradicionales de análisis. Una de sus herramientas más extendidas es la homología persistente, la cual detecta características topológicas relevantes a través de una filtración de complejos simpliciales que puede ser generada a partir de una nube de puntos. Esta técnica de análisis topológico ha ganado una gran relevancia en los últimos años y es aplicada en una amplia variedad de disciplinas científicas.*

*En este trabajo se utilizan las herramientas que nos proporciona la homología persistente en el contexto de la genética para realizar un estudio entre las poblaciones de Coscinasterias tenuispina, una especie de estrella de mar distribuida por diversas zonas del Atlántico y Mediterráneo.*



**Palabras clave:** *Análisis Topológico de Datos – Homología Persistente – Complejo Simplicial – Filtración – Nube de Puntos – Genética – Coscinasterias Tenuispina.*

## *Abstract*

*Topological Data Analysis is a discipline that combines concepts and techniques from topology with data analysis to study and understand the underlying structure in complex data sets. Its objective is to reveal important patterns and features that may be hidden in the data and are not easily detectable using traditional analysis methods. One of its widely used tools is persistent homology, which detects relevant topological features through a filtration of simplicial complexes that can be generated from a point cloud. This technique of topological analysis has gained significant relevance in recent years and it is applied in a wide variety of scientific disciplines.*
*In this work, we use the tools provided by persistent homology in the context of genetics to conduct a study among populations of Coscinasterias tenuispina, a species of starfish distributed in various areas of the Atlantic and Mediterranean.*

**Keywords:** *Topological Data Analysis – Persistent Homology – Simplicial Complex – Filtration – Point Cloud – Genetic – Coscinasterias Tenuispina.*

# Contenido

# Introducción

El Análisis Topológico de Datos (ATD) ha surgido como una técnica novedosa para estudiar y analizar conjuntos de datos complejos en respuesta a la creciente necesidad de desarrollar métodos innovadores y eficientes para el procesamiento de cantidades masivas de datos. En este contexto, la homología persistente se erige como una de las herramientas fundamentales del ATD, proporcionando una perspectiva única que permite explorar la estructura inherente de los datos y revelar patrones complejos y características ocultas. Esta técnica ha encontrado una amplia aplicación en diversos ámbitos científicos. En particular, en el campo de la genética, la homología persistente ofrece una gran cantidad de posibilidades, como la identificación de grupos de individuos con perfiles genéticos similares o la detección de subpoblaciones, entre otros.
Así, el presente trabajo de fin de grado tiene como objetivo principal introducirnos en los principios teóricos de la homología persistente y en el uso práctico de herramientas computacionales para llevarla a cabo y, en última instancia, se propone evaluar el valor añadido que esta técnica puede aportar en un contexto específico relacionado con la genética.

Esta memoria se estructura en tres capítulos, cada uno de ellos enfocado en aspectos clave para comprender y aplicar la homología persistente en diferentes contextos, y en particular, en el ámbito de la genética de poblaciones.

En el primer capítulo se presentarán las nociones básicas de la homología persistente, proporcionando una idea intuitiva de cada una de sus componentes. Se explorarán los conceptos fundamentales de complejos simpliciales, filtraciones y la homología para, finalmente, introducir la noción de persistencia. El resultado del análisis será representado mediante diagramas de persistencia o códigos de barras. Estos nos permitirán identificar características topológicas relevantes de nuestro conjunto de datos tales como las distintas componentes conexas que conforman o agujeros significativos. Si bien no se profundizará en exceso en los detalles matemáticos, se ofrecerá una base sólida para comprender los principios

subyacentes de la homología persistente así como la correspondiente bibliografía a consultar en caso de que el lector quiera profundizar más en los conceptos. Una referencia clásica para el estudio de la homología simplicial es el libro de Munkres [10], mientras que para la homología persistente son referencias básicas los trabajos de Zomorodian y Carlsson [16] y Edelsbrunner, Letscher y Zomorodian [6].

En el segundo capítulo, se introducirá el paquete TDA (*Statistical Tools for Topological Analysis*) [8] para su posterior utilización. A través de ejemplos prácticos, se explorarán las diferentes herramientas que nos proporciona para determinar las características topológicas relevantes en conjuntos de datos de diversa naturaleza. Esto permitirá adquirir una sólida comprensión de cómo aplicar y analizar la homología persistente en el estudio de un conjunto de datos más general.

En el último capítulo, se abordará la aplicación de la homología persistente en el análisis genético de poblaciones de una determinada especie de estrellas de mar llamada *Coscinasterias tenuispina*, con el fin de comprender mejor su estructura genética y las relaciones entre ellas. En primer lugar, se introducirán conceptos genéticos básicos, como los genes, los alelos y microsatélites. A continuación, se explorará la noción de distancia genética utilizando el coeficiente $F_{ST}$, que es una medida de la diferenciación genética entre poblaciones. Finalmente, se emplearán datos reales de estrellas de mar disponibles en [12] con el objetivo de utilizar las herramientas que nos proporciona la homología persistente para analizar las relaciones genéticas entre las distintas poblaciones y extraer conclusiones relevantes sobre la estructura genética y la diversidad de las poblaciones estudiadas.

# 1

# Fundamentos teóricos de la Homología Persistente

En este primer capítulo introduciremos las nociones teóricas básicas de la homología persistente con el objetivo de extraer las características topológicas de un conjunto de datos. Para lograr esto, comenzaremos con una introducción a los complejos simpliciales y filtraciones. En particular, profundizaremos en los complejos de Čech, Vietoris-Rips y alfa, puesto que son ampliamente utilizados en el Análisis Topológico de Datos, y aparecen implementados en casi todos los paquetes de software para el cálculo de la homología persistente. Posteriormente se recordarán conceptos básicos de homología simplicial, para finalizar con la presentación de los diagramas de persistencia y códigos de barras, cuya información será la principal herramienta para conocer la estructura topológica de nuestros datos.
En definitiva, trataremos de exponer de forma intuitiva todos los conceptos base de la homología persistente sin profundizar en su base teórica (no se presentará ninguna demostración). Las referencias básicas para dichas demostraciones y el fundamento teórico de estos conceptos puede consultarse en [2], [4], [10] y [16].

## 1.1. Complejos simpliciales y filtraciones

Con el objetivo de obtener una representación topológica de nuestros datos se introducen los llamados complejos simpliciales. Estos son construcciones basadas en una *nube de puntos* (entendida como un subconjunto finito de $\mathbb{R}^N$ que representará el conjunto de datos) que nos permitirán modelar la estructura geométrica de los datos para obtener una representación de la distribución espacial de los mismos. En general, salvo que se indique lo contrario, consideraremos en $\mathbb{R}^N$ la topología usual (o la inducida por la usual en subconjuntos de $\mathbb{R}^N$).

Para presentar la definición formal de los complejos simpliciales, debemos primero introducir los símplices.

**Definición 1.1.** *Dado* $\{a_0, a_1, \ldots, a_n\}$ *un conjunto geométricamente independiente* [1] $\mathbb{R}^N$*, se define el* ***n-símplice*** *(geométrico) generado por* $\{a_0, a_1, \ldots, a_n\}$*, denotado por* $\sigma = \langle a_0, a_1, \ldots, a_n \rangle$*, como la envolvente convexa de* $\{a_0, a_1, \ldots, a_n\}$*.*

Se dice que $n$ es la **dimensión** de $\sigma$ y $a_0, a_1, \ldots, a_n$ los **vértices** de $\sigma$ (que podemos identificar con puntos de nuestra nube). Instintivamente, podemos ver a los símplices como figuras geométricas básicas de distintas dimensiones, es decir; puntos o vértices (0-símplices), aristas (1-símplices), triángulos rellenos (2-símplices), etc. Se dice que $\tau$ es una **cara** de $\sigma = \langle a_0, a_1, \ldots, a_n \rangle$ si $\tau$ es un símplice generado por un subconjunto no vacío de $\{a_0, a_1, \ldots, a_n\}$ y se denota por $\tau \leq \sigma$.

A partir de estas *'piezas geométricas'* podemos construir los llamados **complejos simpliciales**.

**Definición 1.2.** *Un* ***complejo simplicial*** $K$ *en* $\mathbb{R}^N$ *es una colección de símplices en* $\mathbb{R}^N$ *que cumple las siguientes condiciones*

1. *Si* $\sigma \in K$ *y* $\tau \leq \sigma$*, entonces* $\tau \in K$
2. *Dados* $\sigma_1, \sigma_2 \in K$*, entonces* $\sigma_1 \cap \sigma_2 = \emptyset$ *o bien* $\sigma_1 \cap \sigma_2$ *es una cara común a* $\sigma_1$ *y* $\sigma_2$

A pesar de que esta noción nos permite considerar complejos simpliciales con un número infinito de símplices, nosotros nos limitaremos al caso finito. Dado un complejo simplicial $K$, se define la **dimensión** de $K$ como el máximo de las dimensiones de todos sus símplices. Cabe destacar que un complejo geométrico $K$ se suele representar mediante su poliedro asociado, denotado por $|K|$, que consiste en la unión de todos los símplices de $K$ con la topología inducida por la usual de $\mathbb{R}^N$.

Normalmente resulta más sencillo abstraerse de todas estas nociones geométricas y construir el complejo simplicial de manera abstracta y preocuparnos por su posterior incorporación a un espacio euclídeo más adelante, de esta forma surgen los llamados **complejos simpliciales abstractos**. A diferencia de los complejos simpliciales geométricos, donde los vértices representan puntos del espacio $\mathbb{R}^N$, los símplices de un complejo simplicial abstracto son listas de elementos que pueden representar cualquier cosa.

**Definición 1.3.** *Un* ***complejo simplicial abstracto*** $\mathcal{K}$ *es una colección de subconjuntos no vacíos de un conjunto* $\mathcal{K}_0$ *(conjunto de vértices) tal que:*

1. $\{v\} \in \mathcal{K}$ *para todo* $v \in \mathcal{K}_0$
2. *Si* $\tau \subseteq \sigma$ *con* $\sigma \in \mathcal{K}$*, entonces* $\tau \in \mathcal{K}$

[1] Se dice que $\{a_0, a_1, \ldots, a_n\}$ es geométricamente independiente si los vectores $a_1 - a_0, \ldots, a_n - a_0$ son linealmente independientes

De esta manera, los complejos simpliciales abstractos permiten describir objetos abstractos de manera eficiente y computable. A pesar de sus diferencias, los complejos simpliciales abstractos son equivalentes a los complejos simpliciales geométricos, es decir, ambos pueden utilizarse indistintamente en la mayoría de los casos, ya que dado un complejo simplicial geométrico $K$, podemos pasar al ámbito abstracto mediante el esquema de $K$, $\mathcal{A}(K)$, consistente en la familia de subconjuntos del conjunto de vértices de $K$ que definen un símplice de $K$, y recíprocamente, a partir de un complejo simplicial abstracto $\mathcal{K}$, podemos considerar su realización geométrica, obtenida tras la inclusión de los vértices de cada símplice abstracto en un $n$-símplice geométrico de manera conveniente (véase el Teorema 3.1 de [10] y el Teorema de la realización geométrica de la sección III.1 de [4]). En lo que sigue, trabajaremos con complejos simpliciales abstractos.

*Ejemplo 1.4.* Consideremos los puntos del plano $a_0 = (1, -4)$, $a_1 = (6, 0)$, $a_2 = (4, 4)$ y $a_3 = (-2, 2)$, entonces

$$K = \{a_0, a_1, a_2, a_3, \langle a_0, a_1\rangle, \langle a_1, a_2\rangle, \langle a_1, a_3\rangle, \langle a_0, a_3\rangle, \langle a_2, a_3\rangle, \langle a_1, a_2, a_3\rangle\}$$

es un complejo simplicial geométrico de dimensión 2, y su versión abstracta es

$$\mathcal{A}(K) = \{a_0, a_1, a_2, a_3, \{a_0, a_1\}, \{a_1, a_2\}, \{a_1, a_3\}, \{a_0, a_3\}, \{a_2, a_3\}, \{a_1, a_2, a_3\}\}$$

Ya sea en el ámbito abstracto o geométrico, la representación de dicho complejo simplicial (la de su poliedro asociado) viene dado por la figura 1.1

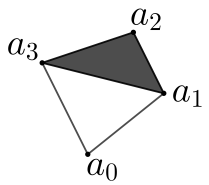


**Figura 1.1.** Poliedro de $K$

**Definición 1.5.** *Dados $K$ y $L$ complejos simpliciales, una* ***aplicación simplicial*** *$\varphi : K \longrightarrow L$, consiste en una aplicación entre sus respectivos conjuntos de vértices verificando que si $a_0, \dots a_n$ son los vértices de un símplice de $K$, entonces $\varphi(a_0), \dots, \varphi(a_n)$ son los vértices de un símplice de $L$. Además, diremos que es un* ***isomorfismo simplicial****, si existe $\psi : L \longrightarrow K$ aplicación simplicial tal que $\psi \circ \varphi = 1_K$ y $\varphi \circ \psi = 1_L$. Donde $1_K$ y $1_L$ denotan las aplicaciones simpliciales identidad definidas entre los vértices de $K$ y $L$ respectivamente.*

Dado un subconjunto finito $S \subset \mathbb{R}^N$ (que representaría nuestros datos como una nube de puntos) y un parámetro $r$ positivo, existen diversas formas de

obtener complejos simpliciales. Las construcciones más habituales en ATD son el complejo de Čech, de Vietoris-Rips y el complejo alfa.

Para definir el complejo de Čech, debemos introducir la noción de **nervio**. Dada $F$ una colección finita de conjuntos en $\mathbb{R}^N$, el **nervio de** $F$ se define como todas las subcolecciones no vacías de $F$ cuyos conjuntos tienen intersección no vacía, es decir,

$$\mathrm{NrV}(F) = \{X \subseteq F / \bigcap_{A \in X} A \neq \emptyset\}$$

Es sencillo comprobar que se trata siempre de un complejo simplicial.

Cabe destacar que existe un teorema con gran relevancia teórica referido al nervio, que enunciamos a continuación

**Teorema 1.6 (Teorema del nervio [1]).** *Sea $F$ una colección finita de conjuntos cerrados convexos en un espacio euclídeo. Entonces, la realización geométrica del nervio de $F$ y la unión de los conjuntos de $F$ son homotópicamente equivalentes.*

En el ámbito del ATD esto resulta de vital importancia ya que podemos estudiar el espacio subyacente a nuestra nube de puntos mediante el complejo simplicial que nos proporciona el nervio de un recubrimiento adecuado.

Con esta idea, si denotamos por $S_r$ a la unión de las bolas cerradas con centro en los puntos de $S$ (nuestro conjunto de datos) y radio $r$, $\bigcup_{x \in S} B(x, r)$, es claro que $\{B(x, r) \setminus x \in S\}$ es un recubrimiento de $S_r$. Entonces se define el **complejo de Čech** asociado a $S$ y $r$, denotado por $\check{C}(r)$, como el nervio de dicho recubrimiento. El Teorema del nervio nos garantiza que $S_r$ y $\check{C}(r)$ tienen el mismo tipo de homotopía.

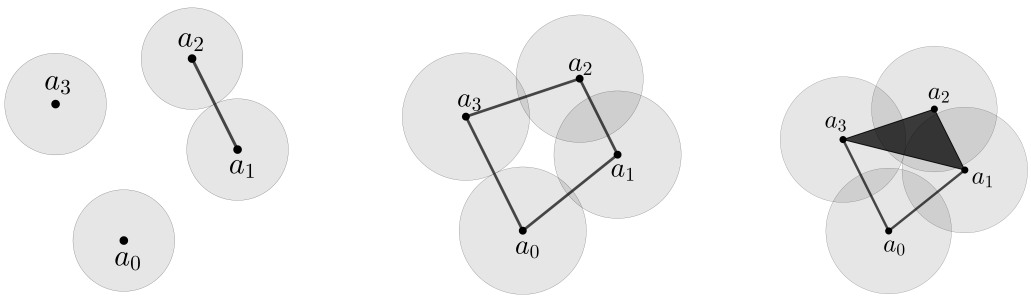


**Figura 1.2.** Complejo de Čech para $r = \frac{4.47}{2}$, $r = \frac{6.71}{2}$ y $r = \frac{8.25}{2}$ de los puntos del ejemplo 1.4

El complejo de **Vietoris-Rips** asociado a $S$ y $r$, se define como sigue

$$\mathrm{VR}(r) := \{\sigma \subseteq S / \operatorname{diam}(\sigma) \leq r\}$$

donde $\operatorname{diam}(\sigma) = \text{máx}\{\mathrm{d}(x, y)/x, y \in \sigma\}$ con d una función distancia cualquiera en $\mathbb{R}^N$. En este capítulo se trabajará con la distancia euclídea salvo que se indique lo contrario.
Es sencillo comprobar que se trata de un complejo simplicial.

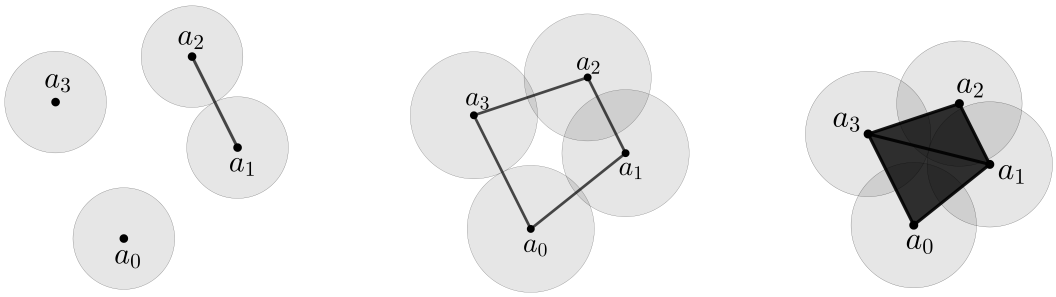


**Figura 1.3.** Complejo de Rips para $r = 4.47$, $r = 6.71$ y $r = 8.25$ de los puntos del ejemplo 1.4

Observamos que en la figura 1.3, los radios de las bolas que se consideran coinciden con los de la figura 1.2 (nótese que en el complejo de Vietoris-Rips el parámetro $r$ hace referencia al diámetro de la bola considerada). Para los dos primeros radios, ambos complejos coinciden, sin embargo cuando el radio de las bolas es $r = \frac{8.25}{2}$, en el complejo de Čech aparece un triángulo con un agujero, mientras que en el de Vietoris-Rips se tiene dicho triángulo relleno. Esto se debe a que las bolas centradas en $a_0$, $a_1$ y $a_3$ no tienen intersección común, luego el símplice $\{a_0, a_1, a_3\}$ no está en el complejo de Čech, sin embargo, el diámetro correspondiente a $\{a_0, a_1, a_3\}$, coincide con el diámetro de la mayor de sus aristas ($\{a_3, a_1\}$), que es exactamente $r$, luego necesariamente, $\{a_0, a_1, a_3\}$ sí se encuentra en el complejo de Rips.
Más adelante veremos que esta diferencia determinará el motivo por cual el complejo de Rips es más utilizado en el conexto del ATD.

Para definir el complejo alfa, debemos introducir antes la noción de la **celda de Voronoi**.

La **celda de Voronoi** de un punto $u$ de $S$, se define como

$$V_u := \{x \in \mathbb{R}^N / \parallel x - u \parallel \leq \parallel x - v \parallel, \text{ para todo } v \in S\}$$

es decir, es el conjunto de puntos de $\mathbb{R}^N$ cuya distancia a cualquier otro punto de $S$ es mayor o igual que su distancia a $u$. En otras palabras, es la intersección de los semiespacios de los puntos que están a igual o menor distancia de $u$ que de $v$, para todo $v$ de $S$. Nótese que geométricamente, podemos constuir estas

celdas considerando los segmentos entre parejas de puntos de $S$ y las rectas perpendiculares a ellos que pasan por el punto medio de cada uno. Dichas rectas nos delimitarán las zonas correspondientes a cada celda de Voronoi.
Así, las celdas de Voronoi son poliedros convexos que recubren todo el espacio. Además, cualquier intersección entre dos celdas se produce en su frontera común ([14] y [15]).
Este concepto se puede visualizar de forma clara cuando trabajamos en el plano, ilustrado en el siguiente ejemplo:

*Ejemplo 1.7.* Consideremos $S = \{A, B, C\} \subseteq \mathbb{R}^2$, donde

$$A = (-5, 0), B = (2, 2) \text{ y } C = (3, -1)$$

Entonces las celdas de Voronoi asociadas a cada punto de $S$, se representan de forma geométrica en la siguiente figura

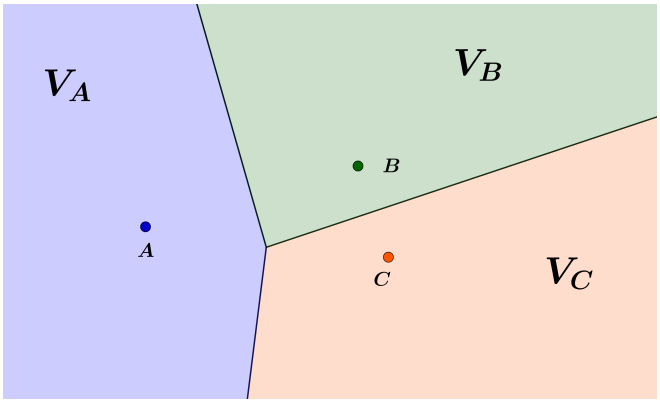


**Figura 1.4.** Celdas de Voronoi de todos los puntos de $S$

La región azul del plano se corresponde con $V_A$, la verde con $V_B$ y la naranja con $V_C$.

Consideremos ahora $B(u, r)$ la bola cerrada con centro $u$ y radio $r$. Al hacer variar $u \in S$, se observa que la unión de estas bolas representa el conjunto de puntos que están como mucho a distancia $r$ de al menos uno de los puntos de $S$. Para descomponer esta unión, intersecamos cada bola con la correspondiente celda de Voronoi, $R_u := B(u, r) \cap V_u$.
Es claro que

$$\bigcup_{u \in S} R_u(r) = \bigcup_{u \in S} B(u, r)$$

pues $\bigcup_{u \in S} V_u = \mathbb{R}^N$, es decir, todos los $R_u(r)$ recubren por completo la unión de las bolas $B(u, r)$. Además, dos elementos de este recubrimiento, o son disjuntos

o se solapan a lo largo de la frontera de una celda de Voronoi. Así, el **complejo alfa** se define como el nervio del recubrimiento de los $R_u$, es decir,

$$Alpha(r) := \{\sigma \subseteq S / \bigcap_{u \in \sigma} R_u(r) \neq \emptyset\}$$

En un principio, este complejo fue introducido por Eldesbrunner, Kirkpatrick y Seidel para puntos del plano $\mathbb{R}^2$ [5] y posteriormente se extendió a $\mathbb{R}^3$ [7]. Es posible considerar otra definición similar al complejo alfa, pero asociando a cada punto $u$ de $S$ un peso $w_u$. En este caso se consideran bolas $B_u$ centradas en $u$ cuyo radio al cuadrado coincide con $w_u$, y consideramos nuevamente la descomposición de la unión de dichas bolas, $R_u = B_u \cap V_u$. Así, de forma análoga al complejo alfa, se define el **complejo alfa pesado** como el nervio de la colección de los $R_u$.
La utilidad de este complejo radica en que podemos considerar bolas de distinto tamaño, esto nos permite en determinados conextos, asociar a cada dato (punto) un valor distintivo que represente alguna propiedad relevante que nos interese destacar. Por ejemplo, en la modelización de biomoléculas, cada átomo es representado por una bola cuyo radio refleja el rango de sus interacciones de van der Waals.
Este último concepto lo introdujo Eldesbrunner en [3] para puntos con pesos en una dimensión genérica fija.

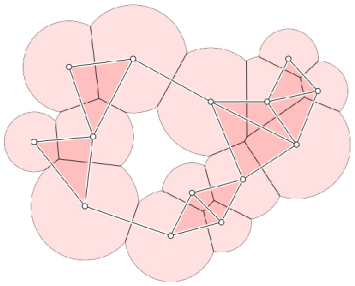

**Figura 1.5.** Figura III.15 de [4] en la que se representa un ejemplo de complejo alfa pesado

*Ejemplo 1.8.* Vamos a calcular el complejo alfa del conjunto $S$ que escogimos en el ejemplo 1.7 eligiendo el parámetro $r = 3$. Para ello, debemos considerar las circunferencias de radio $r$ centradas en cada punto de $S$ y las celdas de Voronoi correspondientes a cada punto, con el fin de calcular $R_A$, $R_B$ y $R_C$, representados en la siguiente figura.

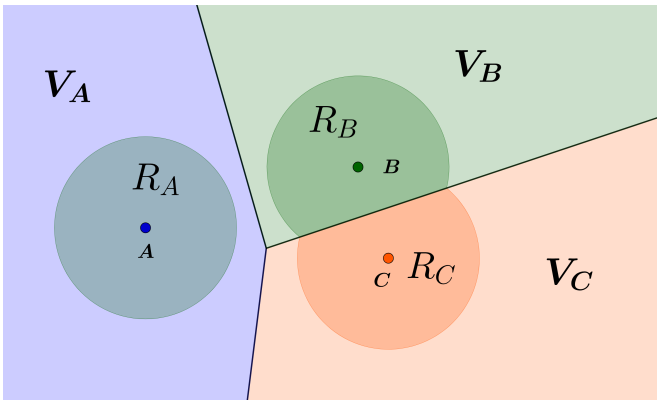


**Figura 1.6.** Representación en el plano de $R_A$, $R_B$ y $R_C$

Entonces, por la propia definición del complejo alpha, obtenemos que $Alpha(3) = \{A, B, C, \{B, C\}\}$


**Figura 1.7.** $Alpha(r = 3)$

Observamos que en los complejos definidos anteriormente, tenemos un parámetro libre $r > 0$ al cual podemos asignarle distintos valores, de esta forma obtenemos una familia de complejos que representarán nuestros datos a distintas escalas. Esto motiva la introducción de las **filtraciones de complejos simpliciales**.
Dado $K$ un complejo simplicial, decimos que $L$ es subcomplejo de $K$ si $L \subseteq K$ y además $L$ es complejo simplicial. Con esta noción podemos definir la **filtración de un complejo simplicial** $K$, esta consiste en una colección anidada de subcomplejos de $K$

$$\emptyset = K_0 \subseteq K_1 \subseteq K_2 \subseteq \cdots \subseteq K_n = K$$

Al variar el parámetro $r$ de los complejos de Čech, Vietoris Rips y alfa, obtenemos las filtraciones basadas en cada uno de ellos.

$$\check{\mathcal{C}}(S) = \{\check{C}(r)\}_{r\geq 0} \quad \mathcal{VR}(S) = \{\mathrm{VR}(r)\}_{r\geq 0} \quad \mathcal{A}lpha(S) = \{Alpha(r)\}_{r\geq 0}$$

donde formalmente se suele considerar que $\check{C}(0) = \mathrm{VR}(0) = Alpha(0) = \emptyset$ (que corresponderían con el $K_0$ de la filtración).

Es sencillo comprobar que se trata de filtraciones de complejos simpliciales.

*Ejemplo 1.9 (Filtración del complejo de Čech).* En la figura 1.2 ilustramos los complejos de Čech de $S = \{a_0, a_1, a_2, a_3\}$ para $r = 4.47/2$, $r = 6.71/2$ y $r = 8.25/2$, en este ejemplo mostramos la filtración completa (figura 1.8), donde $K_1$ se corresponde con $\check{C}(2)$, $K_2$ con $\check{C}(4.47/2)$, $K_3$ con $\check{C}(6.32/2)$, $K_4$ con $\check{C}(6.4/2)$, $K_5$ con $\check{C}(6.71/2)$, $K_6$ con $\check{C}(8.25/2)$, $K_7$ con $\check{C}(4.22)$ y $K_8$ con $\check{C}(8.54/2)$.

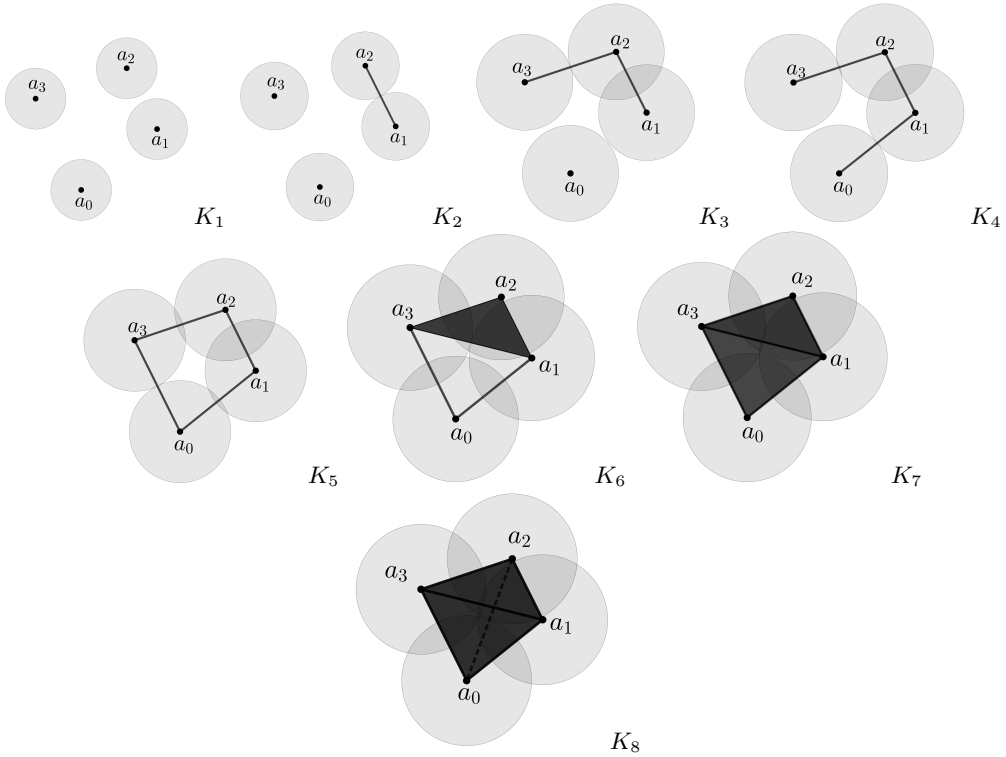


**Figura 1.8.** Ejemplo de filtración de Čech

*Ejemplo 1.10 (Filtración de Vietoris-Rips).* Ilustramos a continuación la filtración completa de Vietoris-Rips de $S = \{a_0, a_1, a_2, a_3\}$ (figura 1.9), donde $K_1$ se corresponde con VR(4), $K_2$ con VR(4.47), $K_3$ con VR(6.32), $K_4$ con VR(6.4), $K_5$ con VR(6.71), $K_6$ con VR(8.25) y $K_7$ con VR(8.54).

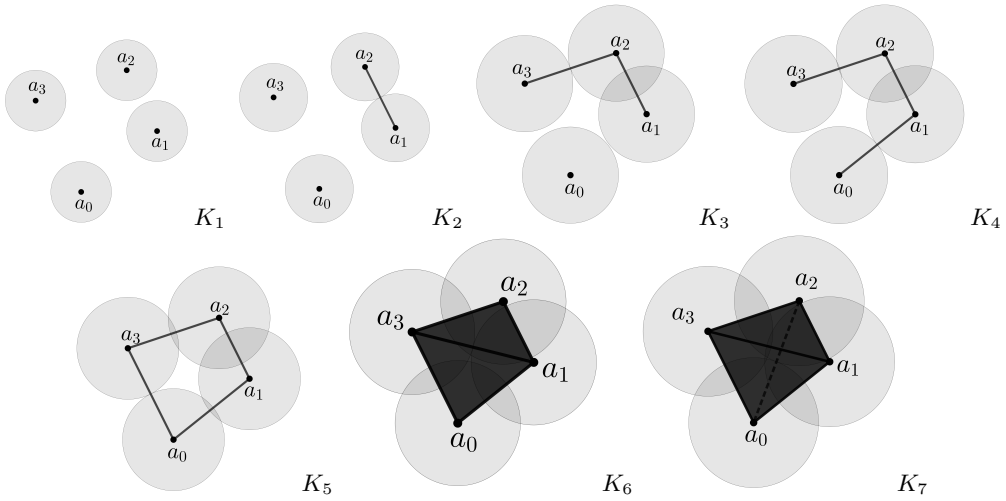


**Figura 1.9.** Ejemplo de filtración de Vietoris-Rips

A través de los dos ejemplos anteriores, se puede apreciar que la filtración basada en el complejo de Čech es más costosa desde un punto de vista computacional que la de Rips. Esto se debe a que la filtración de Čech produce más complejos simpliciales en la filtración, ya que implica tener en cuenta una gran cantidad de intersecciones, lo que resulta particularmente problemático cuando se trabaja con grandes cantidades de datos. Por esta razón, se suele preferir utilizar las filtraciones de Rips en lugar de las de Čech y es por ello que en esta memoria no haremos uso computacional del complejo de Čech. De hecho, la mayoría de los paquetes no hacen uso de las filtraciones basadas en complejos de Čech.

*Ejemplo 1.11 (Filtración del complejo alfa).* Tomamos $S = \{A, B, C\}$ igual que en el ejemplo 1.8. A medida que aumentamos el radio $r$, vamos obteniendo los siguientes complejos representados en 1.10

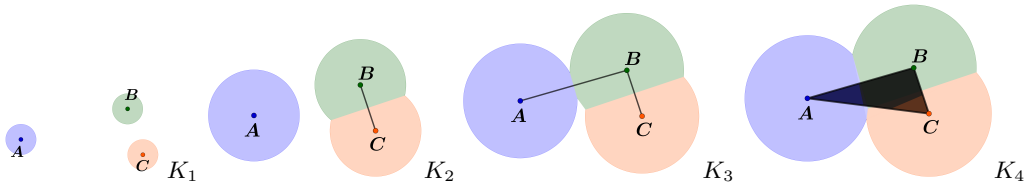


**Figura 1.10.** Ejemplo de filtración alfa

## 1.2. Homología simplicial

En este capítulo no se profundizará en los conceptos de homología que mencionaremos, quedándonos únicamente con la idea genérica e intuitiva. Di-

chas nociones detalladas se pueden consultar en las secciones 2 y 3 [16], aunque la referencia clásica es [10].
En muchas ocasiones, resulta útil conocer las características topológicas del espacio topológico en el que subyace nuestra nube de puntos, tales como el número de componentes conexas o la existencia de agujeros (*holes*) o huecos (*voids*), ya que nos puede proporcionar información valiosa en dos contextos diferentes. Por un lado, si somos capaces de interpretar esta información en el contexto del tipo de datos que manejamos, podría servirnos para describir relaciones existentes entre los distintos puntos (datos) que estudiamos. Por otro lado, puede ser de utilidad para discriminar casos en el procesamiento de datos masivos en estudios comparativos (por ejemplo, con un grupo de control en pruebas médicas).
La topología algebraica nos ofrece para ello una heramienta llamada la homología simplicial asociada a un complejo simplicial $K$ con coeficientes en un cuerpo $\mathbb{F}$. Para cada entero $p$ nos proporciona un objeto algebraico con carácter de $\mathbb{F}$-espacio vectorial. Lo denotamos como $H_p(K;\mathbb{F})$ o bien $H_p(K)$ cuando esté claro el contexto. Sus generadores nos permitirán determinar aquellas características asociadas a nuestro complejo simplicial $K$. En particular, el número de componentes conexas se puede determinar a partir de la dimensión de $H_0(K)$, los agujeros con la dimensión de $H_1(K)$ y los huecos (como el hueco característico de un toro) por la dimensión de $H_2(K)$. En general, $H_p(K)$ será isomorfo a tantas copias de $\mathbb{F}$ como agujeros de dimensión $p$ haya. Si $K = \emptyset$ o bien $p > \dim(K)$, entonces se tendrá que $H_p(K) = 0$.

*Observación 1.12.* Cabe destacar que es posible considerar la homología simplicial con coeficientes en un anillo $R$ cualquiera, obteniéndose (para cada entero $p$) $R$-módulos en lugar de espacios vectoriales. Sin embargo, en la práctica se considera $R = \mathbb{F}$ ya que elimina posibles elementos de torsión y porque permite aplicar un teorema de estructura para $\mathbb{F}[t]$-módulos. Este es una herramienta importante ya que da soporte teórico al algoritmo estándar del cálculo de la homología persistente. No obstante, un análisis detenido de este algoritmo, así como la demostración del teorema, exceden el contenido de esta memoria. Ambos pueden consultarse en [16].

*Ejemplo 1.13.* Dado un cuerpo $\mathbb{F}$ cualquiera, veamos cuáles son los $\mathbb{F}$-espacios vectoriales de homología para cada complejo de la filtración del ejemplo 1.10.

- En el complejo $K_1$, al tener 4 componentes conexas distintas (determinadas por los 4 vértices), se tiene que $H_0(K_1) \cong \mathbb{F}^4$, mientras que $H_p(K_1) \cong 0$ en cualquier otra dimensión, ya que no hay agujeros presentes.
- En $K_2$, los vértices $a_1$ y $a_2$ se conectan formando una única componente conexa. De esta manera, se tiene que $H_0(K_2) \cong \mathbb{F}^3$ y $H_p(K_2) \cong 0$ en cualquier otra dimensión.

- Igual que en el caso anterior, en $K_3$, $a_3$ se conecta con $a_2$, luego pasamos a tener dos componentes conexas distintas y ningún agujero, es decir, $H_0(K_3) \cong \mathbb{F}^2$ y $H_p(K_3) = 0$ para todo $p > 0$.
- En $K_4$, se encuentran conectados todos los vértices, formando así una única componente conexa, lo que implica que $H_0(K_4) \cong \mathbb{F}$. Sin embargo, se sigue teniendo que $H_p(K_4) = 0$ para las demás dimensiones.
- En el caso de $K_5$, ya sabemos que $H_0(K_5) \cong \mathbb{F}$. Además, se observa que aparece un cuadrilátero sin rellenar, luego tenemos un agujero, es decir $H_1(K_5) \cong \mathbb{F}$, sin embargo $H_p(K_5) = 0$ para las dimensiones superiores.
- En $K_6$ se unen los vértices $a_3$ y $a_1$, dividiendo así el cuadrilátero en dos triángulos rellenos coincidentes en una arista. De esta forma el *ciclo* (agujero del cuadrilátero) que *nació* en $K_5$ *muere* en este complejo, pues ahora está relleno. Luego $H_p(K_6) = 0$ para $p > 0$ y es claro que $H_0(K_6) \cong \mathbb{F}$.
- Finalmente, en $K_7$ obtenemos un tetraedro macizo y por tanto tenemos una única componente conexa que carece de agujeros, es decir, $H_0(K_7) \cong \mathbb{F}$ y $H_p(K_7) = 0$ para las demás dimensiones.
  Nótese que si el tetraedro no estuviera relleno, obtendríamos un hueco y por tanto $H_2(K_7) \cong \mathbb{F}$. Sin embargo esto no es posible en una filtración de Vietoris-Rips, pues el diámetro de un tetraedro (3-símplice) coincide con el diámetro de la mayor de sus aristas, luego siempre aparecería relleno (esto ocurre de la misma forma con los triángulos).

Observamos que el comportamiento de estos módulos de homología los podríamos representar mediante el siguiente diagrama

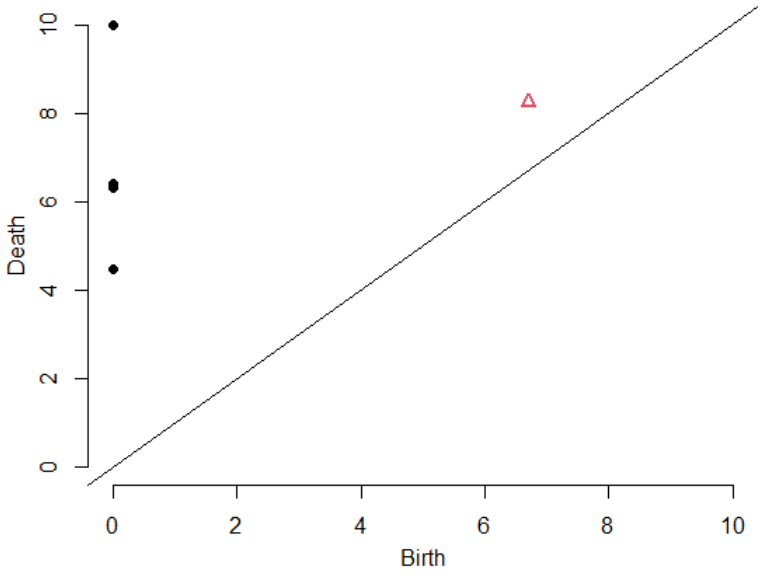


**Figura 1.11.** Diagrama de persistencia de la filtración del ejemplo 1.10

Donde los puntos hacen referencia a las componentes conexas y los triángulos a los agujeros. Un punto $(i, j)$ del diagrama indicaría que aparece (nace) una componente conexa en el complejo $\mathrm{VR}(r = i)$ y desaparece (muere) en el complejo $\mathrm{VR}(j)$. De manera similar, un triángulo situado en $(i, j)$ indica la aparición de un agujero en el complejo $\mathrm{VR}(i)$ y su desaparición en $\mathrm{VR}(j)$. Obtenemos así de forma gráfica la persistencia de estas componentes o agujeros, cuya definición formal daremos en la siguiente sección.

## 1.3. Homología persistente

La homología persistente surge con la motivación de analizar cómo cambia la homología a través de una filtración de complejos simpliciales. Por ejemplo, para estudiar comportamientos como los que observábamos en el ejemplo 1.13 y analizar la persistencia de los agujeros y componentes mediante los diagramas de persistencia (figura 1.13). Para computar la homología persistente y poder obtener diagramas de persistencia, se tiene el llamado **algoritmo estándar**, que fue introducido con coeficientes en el cuerpo $\mathbb{F}_2$ (el cuerpo con dos elementos) en [4] y extendido para cuerpos genéricos en [16]. Posteriormente se han desarrollado otros algoritmos y variantes cuya salida computacional es esencialmente la misma. En esta sección trataremos de exponer las nociones clave de la

homología persistente para su posterior utilización en capítulos siguientes.

Partimos de una filtración de complejos simpliciales (por ejemplo, basada en los complejos de Vietoris-Rips o alfa)

$$\emptyset = K_0 \subseteq K_1 \subseteq \cdots \subseteq K_n = K.$$

Para cada $i \leq j$ tenemos aplicaciones simpliciales inclusión $\zeta^{i,j} : K_i \hookrightarrow K_j$, que inducen homomorfismos de $R$-módulos $f_p^{i,j} : H_p(K_i) \to H_p(K_j)$, definidos como $f_p^{i,j}([z]) := [\zeta^{i,j}(z)]$ para cada dimensión $p$ (ver sección 2.6 de [16] y sección 3 de [6]). Así, obtenemos la siguiente sucesión de $R$-módulos de homología

$$0 = H_p(K_0) \xrightarrow{f_p^0} H_p(K_1) \xrightarrow{f_p^1} \ldots \xrightarrow{f_p^{n-1}} H_p(K_n) = H_p(K) \tag{1.1}$$

para cada dimensión $p$, donde $f_p^i := f_p^{i,i+1}$ para todo $i \in \{0, 1, \ldots, n-2\}$. Observamos que en la transición de $K_{i-1}$ a $K_i$ pueden surgir nuevas clases de homología y desaparecer otras, ya sea porque se vuelven triviales o porque se relacionan con otras clases existentes.

**Definición 1.14.** *Se define el* $p$***-ésimo*** $R$***-módulo de persistencia*** *como las imágenes de los homomorfismos inducidos por las inclusiones*

$$H_p^{i,j} = \mathrm{Im}(f_p^{i,j}), \text{ para } 0 \leq i \leq j \leq n$$

*Denotaremos la imagen de* $f_p^i$ *por* $H_p^i := \mathrm{Im}(f_p^i)$.

**Definición 1.15.** *Sea* $\alpha$ *una clase en* $H_p(K_i)$*, decimos que* $\alpha$ ***nace*** *en* $K_i$ *si* $\alpha \notin H_p^{i-1,i}$ *y en ese caso, decimos que* ***muere entrando*** *en* $K_j$ *si se fusiona con una clase anterior en la transición de* $K_{j-1}$ *a* $K_j$*, es decir,* $f_p^{i,j-1}(\alpha) \notin H_p^{i-1,j-1}$ *pero* $f_p^{i,j}(\alpha) \in H_p^{i-1,j}$*. En este caso se dice que el* ***índice de persistencia*** *de* $\alpha$ *es* $j - i$*. Si* $\alpha$ *nunca muere, se dice que su índice de persistencia es infinito.*

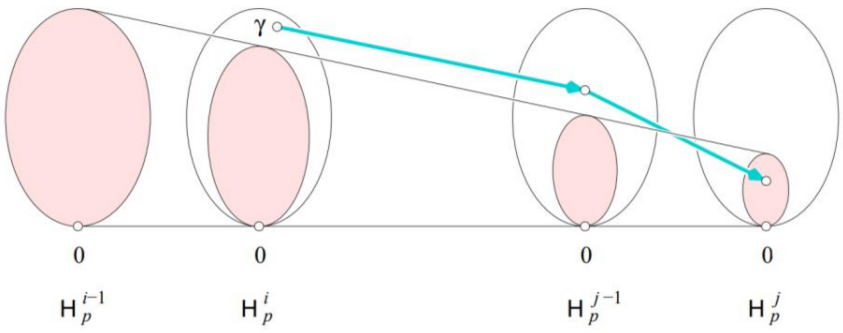


**Figura 1.12.** Figura VII.2 de [4]. La clase $\gamma$ nace en $K_i$ pues no se encuentra en $H_p^i$ (zona sombreada), es decir, no cae en $f_p^{i-1}(H_p(K_{i-1}))$. Además, muere entrando en $K_j$ pues su imagen $f_p^{i,j}(\gamma)$ entra por primera vez en la imagen de $H_p(K_{i-1})$ justo en $H_p^j$

Una forma de ilustrar los conceptos anteriores es mediante los **diagramas de persistencia**. Estos representan el nacimiento y la muerte de las clases generadoras de los $R$-módulos de homología en la sucesión mostrada en 1.1. Si $\alpha$ nace en $K_i$ y muere en $K_j$, se le asigna el punto del plano $(i, j)$. Cabe destacar que como $j > i$, estos puntos siempre estarán por encima de la diagonal y cuanto más alejados estén de esta, mayor será su persistencia en la filtración. Los puntos que estén muy cercanos a la diagonal (índice de persistencia pequeño) se considerarán 'ruido', ya que estos no tienen una persistencia significativa en la filtración y por tanto no nos darán información relevante en cuanto a nuestra nube de puntos.

En la figura 1.13 se mostró el diagrama de persistencia asociado a la filtración de Vietoris-Rips presentada en el ejemplo 1.10, cuyos módulos de homología estudiamos en el ejemplo 1.13. Los puntos representan las componentes conexas (es decir, las clases generadoras del $R$-módulo de homología en dimensión 0) y los triángulos representan los ciclos (clases generadoras en dimensión 1, es decir, agujeros en dimensión 1). Podemos observar que las componentes conexas se van uniendo hasta formar una sola (el punto más alto indica que existe una componente conexa que persiste hasta el final de la filtración), mientras que hubo un ciclo con poca persistencia en la filtración (el triángulo situado en (6.71,8.25), que hace referencia al cuadrilátero que apareció en $K_5 = \mathrm{VR}(6.71)$ y murió en $K_6 = \mathrm{VR}(8.25)$).

Existe una forma alternativa de representar la persistencia de dichas características topológicas mediante los llamados **códigos de barras**. Estos se basan en la misma idea que los diagramas de persistencia pero se diferencian en que utiliza segmentos para representar las etapas de nacimiento y muerte de las clases generadoras.

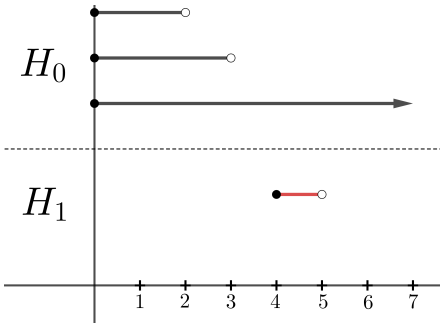


**Figura 1.13.** Código de barras asociado a la filtración del ejemplo 1.10. En negro se representa la persistencia de las componentes conexas y en rojo la persistencia de los agujeros de dimensión uno

# 2

# Una implementación para R: el paquete TDA

En la actualidad existe una amplia gama de implementaciones de los algoritmos citados, y paquetes diseñados para el cómputo de la homología persistente [11], disponibles en distintos lenguajes de programación. En nuestro caso particular, hemos optado por utilizar el paquete TDA (*Statistical Tools for Topological Analysis*) [8] que nos proporciona una serie de funciones de R permitiéndonos aplicar algoritmos eficientes de librerías de C++, tales como *GUDHI* [18], *Dionysus* [19] y *PHAT* [20], enfocadas al cálculo de la homología persistente. Con ellas y las distintas funciones que nos facilita el paquete, podremos ser capaces (en particular) de calcular las filtraciones de Vietoris-Rips y Alfa de una nube de puntos dada y de obtener su diagrama de persistencia asociado.
En este capítulo introduciremos algunos ejemplos prácticos a partir de datos artificiales utilizando diversas funciones de TDA. De esta manera, podremos comprender cómo utilizar estas herramientas de forma efectiva y extraer conclusiones significativas.
Al tratarse de ejemplos con nubes de muy pocos puntos, no serán relevantes la elección del algoritmo ni del tipo de librería empleada. En cualquier caso, un estudio comparativo de las características y rendimiento de las librerías *GUDHI*, *Dionysus*, y *PHAT* (además de algunas otras no implementadas en el paquete TDA), puede consultarse en la sección 7 de [11].

## 2.1. Filtraciones Vietoris-Rips

El paquete nos proporciona la función `ripsDiag` para el cálculo del diagrama de persistencia de una filtración de Rips construida a partir de una nube de puntos dada. Sus argumentos son los siguientes, con indicación de los valores por defecto cuando los hay:

```
ripsDiag(X, maxdimension, maxscale, dist = "euclidean",
library = "GUDHI", location = FALSE, printProgress = FALSE)
```

Explicaremos cada uno mediante el siguiente ejemplo:

*Ejemplo 2.1 (Diagrama de persistencia de la filtración 1.9).* Recordemos que en esta filtración partíamos de la nube de puntos $S = \{a_0, a_1, a_2, a_3\}$, donde $a_0 = (1, -4), a_1 = (6, 0), a_2 = (4, 4)$ y $a_3 = (-2, 2)$.
El parámetro `X` vendrá dado por dicha nube de puntos, debe ser una matriz que recoja los puntos por filas.

```
> library(TDA)    # Se carga la librería TDA
> S<-cbind(c(1,6,4,-2), c(-4,0,4,2))    # Matriz de nube de puntos
```

Podemos especificar la máxima dimensión de las clases generadoras que aparecerán en el diagrama mediante el argumento `maxdimension` (0 para componentes conexas, 1 para agujeros , etc.) Por otro lado, con `maxscale` fijamos el valor máximo de $r$ que se considerará en la filtración.

```
> maxdimension <- 2      # Componentes, agujeros y huecos
> maxscale <- 10         # Se tomará como máximo valor r=10
```

Finalmente generamos el diagrama de persistencia

```
> DiagRips <- ripsDiag(X=S,  maxdimension, maxscale,
+                      dist = 'euclidean', library = 'Dionysus',
+                     location = TRUE, printProgress = TRUE)
```

En este caso hemos utilizado la librería `"Dionysus"`. Alternativamente, podríamos haber usado la librería `"GUDHI"`, además también es posible proporcionarle un vector de caracteres de longitud dos donde el primer elemento especifica la librería utilizada para computar la filtración de Rips (se puede utilizar tanto `"Dionysus"` como `"GUDHI"`) y el segundo caracter indica la librería que calcula el diagrama de persistencia (también pueden ser utilizada cualquiera de las dos librerías anteriores y adicionalmente, `"PHAT"`).
El argumento `printProgress` nos muestra una barra de progreso y el tiempo en la ejecución de la función así como el número total de símplices que resultaron en la filtración

```
# Generated complex of size: 15

0%   10   20   30   40   50   60   70   80   90   100%
|----|----|----|----|----|----|----|----|----|----|
**************************************************
# Persistence timer: Elapsed time [ 0.001000 ] seconds
```

La función `ripsDiag` nos devuelve una lista (guardada en el ejemplo como `DiagRips`) con los elementos `diagram`, `birthLocation`, `deathLocation` y `cycleLocation` (estos tres últimos debido a que `location=TRUE` y `library="Dionysus"`, también es posible utilizar `"PHAT"` para obtener `birthLocation` y `deathLocation`, pero `"Dionysus"` es indispensable para obtener `cycleLocation`).

El elemento `diagram` nos devuelve una matriz cuya primera columna contiene la dimensión de cada clase generadora (0 para componentes, 1 para agujeros, etc.) y la segunda y tercera columna son el nacimiento y muerte de dichas clases.

```
> DiagRips$diagram    # Accedemos al elemento diagram de DiagRips
     dimension    Birth     Death
[1,]         0 0.000000 10.000000
[2,]         0 0.000000  6.403124
[3,]         0 0.000000  4.472136
[4,]         0 0.000000  6.324555
[5,]         1 6.708204  8.246211
```

Con esta información, podemos representar el diagrama de persistencia

```
> # Gráfico del diagrama de persistencia
> plot(DiagRips[['diagram']])
```

Obteniendo la figura 1.13 presentada en el primer capítulo.

Los elementos `birthLocation` y `deathLocation` son dos matrices que nos proporcionarán información sobre el nacimiento y muerte de cada clase generadora representada en el diagrama de persistencia.
En la fila i-ésima de `birthLocation` se representa el punto que da lugar al nacimiento de la clase generadora representada en la fila i-ésima de `diagram` y que muere con el punto de la fila i-ésima de `deathLocation`.

```
> DiagRips$birthLocation
     [,1] [,2]
[1,]    1   -4
[2,]    6    0
[3,]    4    4
[4,]   -2    2
[5,]    1   -4
>
> DiagRips$deathLocation
     [,1] [,2]
[1,]    1   -4
[2,]    1   -4
[3,]    6    0
[4,]    4    4
[5,]    6    0
```

Obtenemos que la componente conexa que nunca muere es el punto $(1, -4)$. Por otro lado, la componente conexa formada por el punto $(6, 0)$ muere cuando se conecta con $(1, -4)$ en el valor 6.403124 de la filtración, de la misma forma,

$(4,4)$ muere cuando se conecta con $(6,0)$ y $(-2,2)$ cuando se conecta a $(4,4)$, estos últimos en los valores 4.472136 y 6.324555 de la filtración.
En el caso de los ciclos, el 1-ciclo representado en el diagrama se inicia con el punto $(1,-4)$, pues este se une a $(-2,2)$ cerrando el ciclo (ver figura 1.9), mientras que muere en $(6,0)$, pues este punto se conecta con $(-2,2)$ y en consecuencia se crean dos triángulos rellenos que 'matan' al hueco anterior.

`cycleLocation` es una lista de longitud $P$ donde $P$ es el número de puntos que aparecen en el diagrama, en nuestro caso, es una lista de longitud 5. Cada elemento nos indicará los vértices con los que se ha formado cada componente conexa o cada ciclo mostrado en el diagrama.

```
> DiagRips$cycleLocation[[1]]
, , 1



, , 2
```

obtenemos un array vacío, esto hace referencia a la componente conexa del diagrama que nace y nunca muere.
Si miramos el segundo elemento (que corresponde con la segunda fila de `DiagRips$diagram`)

```
> DiagRips$cycleLocation[[2]]
, , 1

     [,1]
[1,]    6
[2,]    1

, , 2

     [,1]
[1,]    0
[2,]   -4
```

En `DiagRips$cycleLocation[[2]][, ,1]` obtenemos las primeras coordenadas de los puntos (dispuestos por filas), mientras que en `DiagRips$cycleLocation[[2]][, ,2]` se muestran las segundas coordenadas, es decir, se representa la unión del vértice $(6,0)$ con $(1,-4)$. Para observar esto de forma más cómoda se puede proceder como sigue

```
> DiagRips$cycleLocation[[2]][, ,]
     [,1] [,2]
```

```
[1,]    6    0
[2,]    1   -4
```

De esta forma podemos visualizar de forma clara los vértices que están interviniendo.
Si ahora accedemos al elemento número 5 (correspondiente con el ciclo de dimensión 1 que aparece en el diagrama de persistencia)

```
> DiagRips$cycleLocation[[5]] # Ciclo de dimensión 1
, , 1

     [,1] [,2]
[1,]    1   -2
[2,]    1    6
[3,]    4   -2
[4,]    6    4

, , 2

     [,1] [,2]
[1,]   -4    2
[2,]   -4    0
[3,]    4    2
[4,]    0    4
```

Obtenemos que el ciclo de dimensión 1 se formó mediante la unión de aristas generadas por $(1,-4)$ y $(-2,2)$, $(1,-4)$ y $(6,0)$, $(4,4)$ y $(-2,2)$ y finalmente, $(6,0)$ y $(4,4)$.

Cabe destacar que en este ejemplo hemos utilizado la distancia euclídea `dist="euclidean"`, sin embargo, también es posible indicar `dist="arbitrary"`, en este caso, el argumento `X` debe ser una matriz de distancias $n \times n$ donde $n$ es el número de puntos que se esté considerando o bien un objeto distancia proporcionado por el comando `dist` del paquete `stats`, presente en toda instalación de `R`.

Si quisiéramos reproducir la filtración anterior pero con `dist="arbitrary"`, generamos la correspondiente matriz de distancias (en este caso con la euclídea).

```
> S_dist <- dist(S)     # Matriz de distancias
> S_dist
         1        2        3
2 6.403124
3 8.544004 4.472136
4 6.708204 8.246211 6.324555
```

Cada fila y columna está etiquetada según los puntos entre los que se está calculando su distancia. Por ejemplo el primer elemento de la matriz se corresponde con la distancia entre el punto `1` (el $(1, -4)$) por el orden de los puntos en la matriz `S`) y el punto `2` (el $(6, 0)$).

```
> DiagRips_dist <- ripsDiag(X=S_dist,  maxdimension, maxscale,
+                           dist = 'arbitrary',
+                           library = 'Dionysus', location = TRUE,
+                           printProgress = TRUE)
> DiagRips_dist$diagram
     dimension Birth Death
[1,]         0  0.00 10.00
[2,]         0  0.00  6.40
[3,]         0  0.00  4.47
[4,]         0  0.00  6.32
[5,]         1  6.71  8.25
```

Obtenemos el mismo diagrama de persistencia que el caso anterior.
Accediendo a los elementos `birthLocation` y `deathLocation` obtenemos esencialmente la misma información pero se hace referencia a un punto por un único valor, el cual se corresponde a la etiqueta que está usando la matriz de distancias para cada punto.

```
> DiagRips_dist$birthLocation
[1] 1 2 3 4 1
> DiagRips_dist$deathLocation
[1] 1 1 2 3 2
```

En este caso, el número `i` hace referencia al punto $a_{i-1}$ para $i \in \{1, \ldots, 4\}$.
En `cycleLocation` ocurre exactamente lo mismo, obtenemos la información resultante con `dist="euclidean"` pero cada punto se representa por los valores mencionados anteriormente.

Es evidente que en el caso que estamos considerando, el uso de la matriz de distancias resulta innecesario, ya que `dist="euclidean"` nos permite obtener el diagrama de persistencia sin tener que calcular la distancia entre cada punto. Sin embargo, esta alternativa resulta de gran utilidad cuando disponemos de datos que no pueden ser relacionados de manera adecuada mediante dicha distancia, y en su lugar se requiere de otras medidas de distancia, como por ejemplo, en el caso de disponer de información genética de distintos individuos y requerir del uso de la distancia genética (como veremos en el siguiente capítulo), o también si tuviéramos datos que requirieran distancias como la del máximo, Manhattan, Canberra, binaria o Minkowski, las cuales podemos calcular mediante el comando `dist`.

Una vez introducidos los conceptos básicos de la función, veamos cómo esta nos proporciona información relevante respecto a nuestros datos en este ejemplo. Partíamos de la siguiente nube de puntos

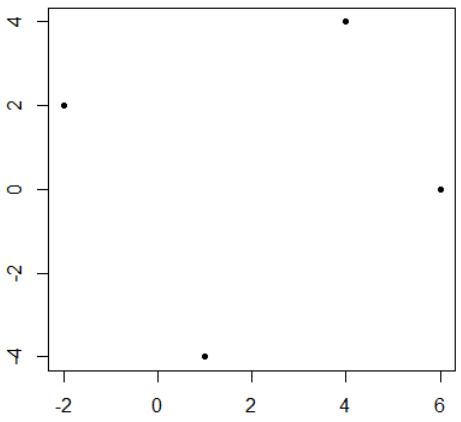


**Figura 2.1.** Nube de puntos S

Es posible intuir que dichos puntos son el borde de un agujero, lo cual justifica la aparición de un ciclo en el diagrama de persistencia 1.13. No obstante, este ciclo se encuentra en una posición muy cercana a la diagonal, indicando una baja persistencia, pudiendo incluso considerarse 'ruido'.
Veamos qué ocurre si añadimos más puntos a S de forma que sea más evidente la presencia de un agujero.

```
> nuevos_puntos <- cbind(c(3.5,5,1,-0.5), c(-2,2,3,-1))
> S_ <- rbind(S, nuevos_puntos)
> S_
     [,1] [,2]
[1,]  1.0   -4
[2,]  6.0    0
[3,]  4.0    4
[4,] -2.0    2
[5,]  3.5   -2
[6,]  5.0    2
[7,]  1.0    3
[8,] -0.5   -1
> plot(S_, pch=20, asp=1, xlab='', ylab='')
>
```

```
> DiagRips_ <- ripsDiag(X=S_,  maxdimension, maxscale,
+                           dist = 'euclidean',library = 'Dionysus',
+                           location = TRUE, printProgress = TRUE)
> plot(DiagRips_[['diagram']])
```

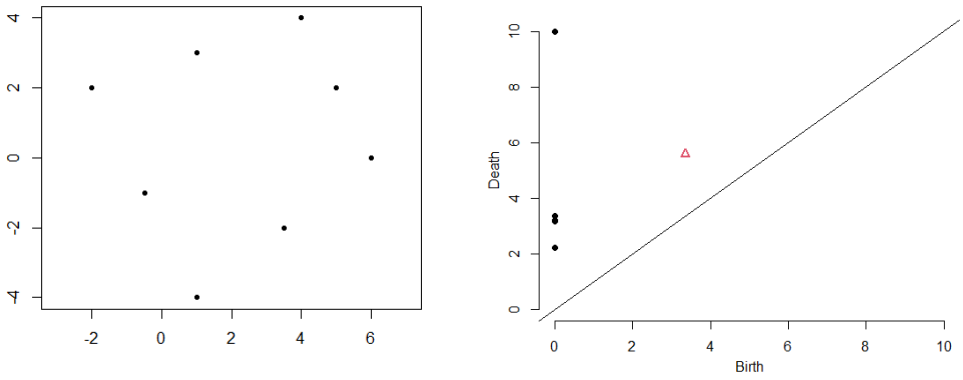


**Figura 2.2.** Nueva nube de puntos a la izquierda y su diagrama de persistencia a la derecha

Se observa un mayor índice de persistencia que en el caso anterior.

```
> DiagRips_$diagram
      dimension    Birth     Death
 [1,]         0 0.000000 10.000000
 [2,]         0 0.000000  3.201562
 [3,]         0 0.000000  2.236068
 [4,]         0 0.000000  3.162278
 [5,]         0 0.000000  3.201562
 [6,]         0 0.000000  2.236068
 [7,]         0 0.000000  3.162278
 [8,]         0 0.000000  3.354102
 [9,]         1 3.354102  5.590170
```

Tomando `S` como nube de puntos obteníamos un índice de persistencia de

```
> 8.246211-6.708204
[1] 1.538007
```

mientras que con `S_`

```
> 5.590170-3.354102
[1] 2.236068
```

Si afinamos aún más nuestra nube de puntos, obtendremos que el ciclo tiene un índice de persistencia mayor (el triángulo rojo aparecerá más alejado de la diagonal), pues resultará más evidente el agujero

```
> nuevos_puntos <- cbind(c(-0.5,2.5,4.5,5.5,4.75,2.25,0.25,-1.25),
+                        c(2.5,3.5,3,1,-1,-3,-2.5,0.5))
> S_f <- rbind(S_, nuevos_puntos)
> plot(S_f, pch=20, asp=1, xlab='', ylab='')
>
> DiagRips_f <- ripsDiag(X=S_f,  maxdimension, maxscale,
+                        dist = 'euclidean',library = 'Dionysus',
+                        location = TRUE, printProgress = TRUE)
> plot(DiagRips_f[['diagram']])
```

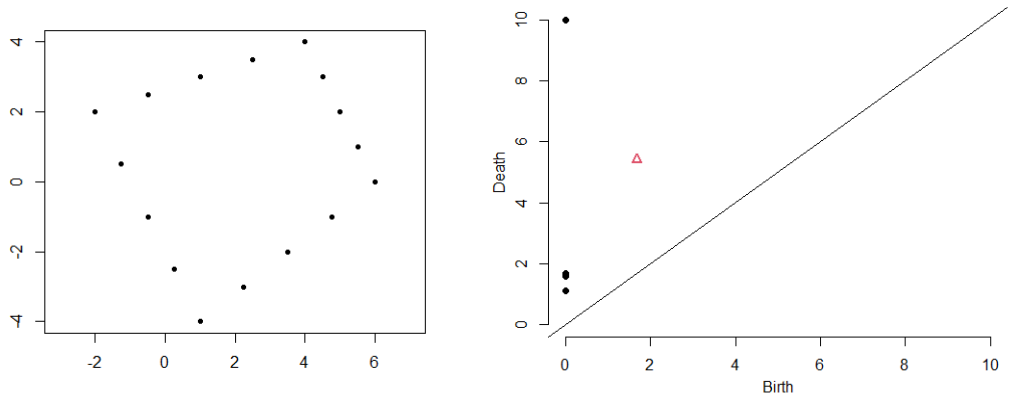


**Figura 2.3.** Nube de puntos S_f a la izquierda y su diagrama de persistencia a la derecha

Cabe destacar que la función `plot`, nos permite realizar modificaciones en dicho diagrama, tales como los límites de los ejes coordenados (`diagLim`), elegir una única dimensión a representar (`dimension`), y considerar el código de barras en lugar del diagrama de persistencia (`barcode`). A continuación representamos los códigos de barras asociados a las nubes de puntos `S`, `S_` y `S_f`

```
> par(mfrow = c(3,1))
> plot(DiagRips$diagram, diagLim = NULL,
+      dimension = NULL, barcode = TRUE)
> title("S")
> plot(DiagRips_$diagram, diagLim = NULL,
+      dimension = NULL, barcode = TRUE)
> title("S_")
```

```
> plot(DiagRips_f$diagram, diagLim = NULL,
+      dimension = NULL, barcode = TRUE)
> title("S_f")
```

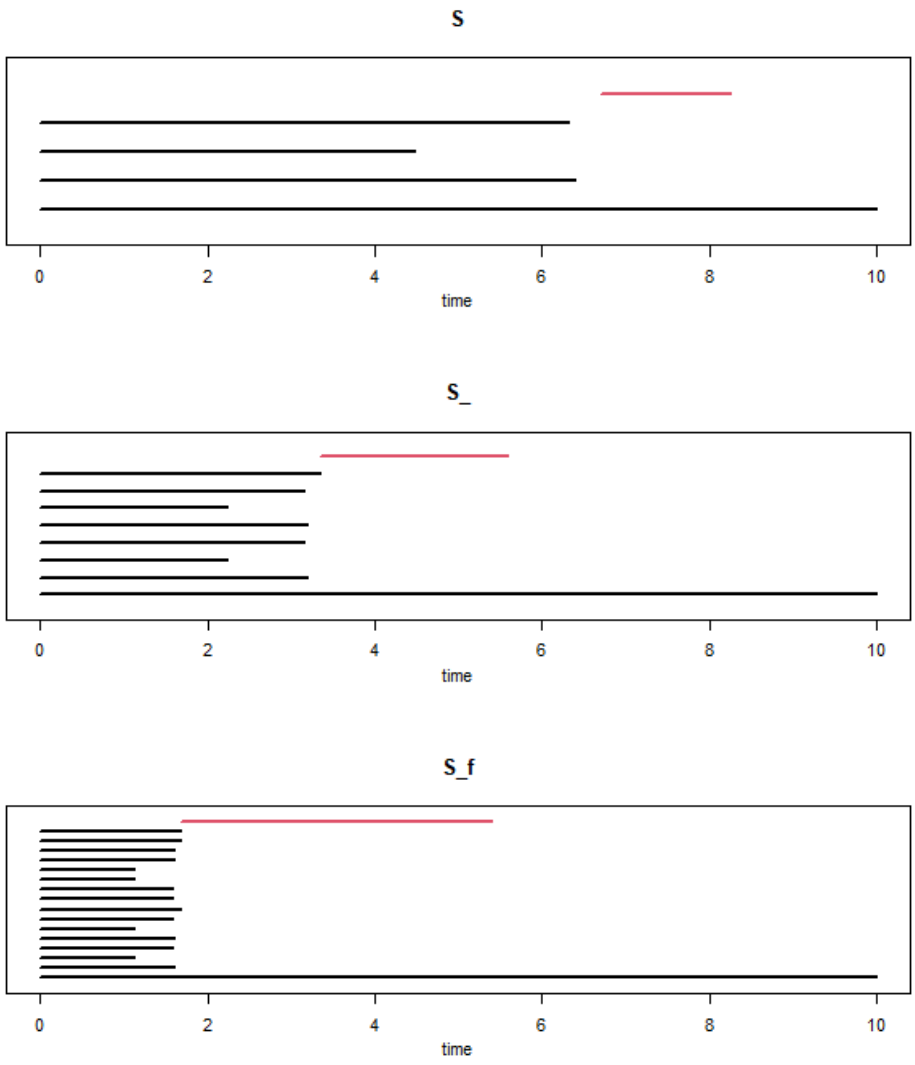


**Figura 2.4.** Códigos de barras de `S`, `S_` y `S_f`. En rojo se representan los agujeros y en negro las componentes conexas

Con esta forma alternativa de representar la persistencia, se observa de manera clara que al agregar más puntos y hacer más evidente el agujero, se obtiene una persistencia mayor. Por esta razón, los códigos de barras resultan útiles, especialmente cuando se busca comparar la persistencia entre diferentes casos.

## 2.2. Filtraciones alfa

De forma análoga al caso de Vietoris-Rips, el paquete TDA nos proporciona la función `alphaComplexDiag` para el cómputo del diagrama de persistencia de la filtración alfa. A continuación se muestran los diferentes argumentos, con sus valores por defecto cuando los tienen

```
alphaComplexDiag(X, maxdimension = NCOL(X) - 1, library = "GUDHI",
location = FALSE, printProgress = FALSE)
```

La principal diferencia respecto al caso de Rips es que no disponemos del parámetro `dist` como alternativa a utilizar la distancia euclídea y tampoco del argumento `maxscale`. Por otro lado, también difiere un poco en las librerías que hay que considerar (ver la documentación de la función `alphaComplexDiag` de [8]).

*Ejemplo 2.2 (Diagrama de persistencia de la filtración 1.10).* Partíamos de $S = \{(-5, 0), (2, 2), (3, -1)\}$

```
> library(TDA)
>
> S <- cbind(c(-5,2,3),c(0,2,-1))      # Nube de puntos
>
> DiagAlphaCmplx <- alphaComplexDiag(
+                   S, library = c("GUDHI", "Dionysus"),
+                   location = FALSE, printProgress = FALSE)
> plot(DiagAlphaCmplx[["diagram"]])
```

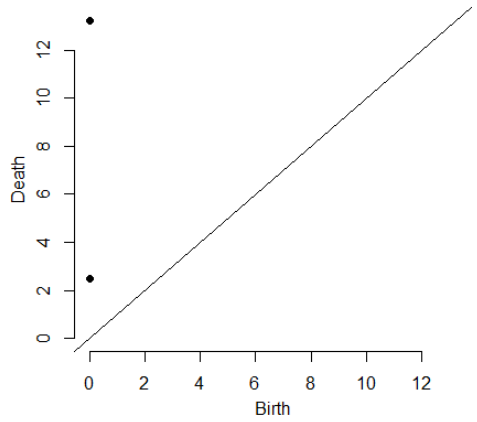


**Figura 2.5.** Diagrama de persistencia de $S$

Al no poder ajustar el parámetro `maxscale`, la figura 2.6 no nos está proporcionando la información completa de la filtración, pues como podíamos observar en la figura 1.10, se produce el nacimiento y muerte de dos componentes conexas mientras que una última persiste hasta el final. Este hecho lo podemos comprobar accediendo al elemento `diagram` de `DiagAlphaCmplx`

```
> DiagAlphaCmplx$diagram
     dimension Birth Death
[1,]         0     0   Inf
[2,]         0     0 13.25
[3,]         0     0  2.50
```

La componente conexa que muere en `13.25` no se está mostrando, esto lo podemos corregir aumentando el rango de los ejes en las opciones de la función `plot`

```
> plot(DiagAlphaCmplx$diagram, diagLim = c(0,14), dimension = NULL,
+      col = NULL, rotated = FALSE, barcode = FALSE)
```

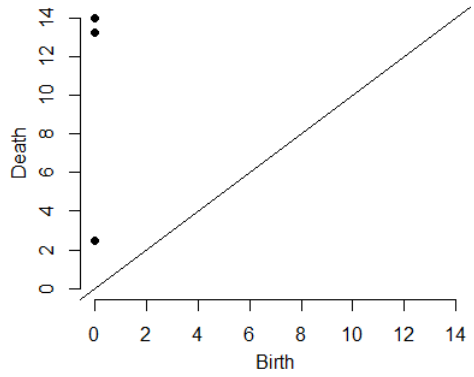


**Figura 2.6.** Diagrama de persistencia de $S$

Cabe destacar que para obtener más información de la filtración podemos utilizar la función `alphaComplexFiltration` (también tenemos la versión análoga para el caso de Rips: `ripsFiltration` [8]).

```
alphaComplexFiltration(X, library = "GUDHI", printProgress = FALSE)
```

Esta función nos devuelve una lista con los elementos: `cmplx` y `values` (entre otros). Aplicado a nuestro caso:

```
> AlphaFilt <- alphaComplexFiltration(
+   S, library = "GUDHI", printProgress = FALSE)
```

`cmplx` es una lista cuyo i-ésimo elemento representa los vértices del i-ésimo símplice de la filtración

```
> AlphaFilt$cmplx[[1]] # Vértices del primer símplice de la lista
[1] 1
```

El primer símplice de la filtración se corresponde con el punto $(-5, 0)$ (por el orden de la matriz de coordenadas). Así, Los 3 primeros símplices de la lista se corresponderán con los puntos de $S$.

```
> AlphaFilt$cmplx[[4]] # Vértices del cuarto símplice de la lista
[1] 3 2
```

El cuarto símplice de la lista se corresponde con el símplice de vértices $(3, -1)$ y $(2, 2)$.
De forma análoga obtenemos los demás símplices de la lista

```
> AlphaFilt$cmplx[[5]]
[1] 2 1
> AlphaFilt$cmplx[[6]]
[1] 3 1
> AlphaFilt$cmplx[[7]]
[1] 3 2 1
```

El argumento `values` nos indica el valor en el que apareció el i-ésimo símplice en la filtración

```
> AlphaFilt$values[[4]]
[1] 2.5
> AlphaFilt$values[[5]]
[1] 13.25
> AlphaFilt$values[[6]]
[1] 16.28072
> AlphaFilt$values[[7]]
[1] 16.28072
```

Así, el símplice de vértices `3 2` aparece cuando $r = 2.5$, el `2 1` cuando $r = 13.25$ y los dos últimos cuando $r = 16.28072$.

## 2.3. Ejemplos sintéticos

A continuación, mostraremos varios ejemplos en los que calcularemos el diagrama de persistencia de conjuntos de puntos que presentan una distribución espacial de relevancia topológica (ver [21] para consultar los scripts del código que se muestra a continuación).

*Ejemplo 2.3 (Anillo).*

```
> x <- runif(1000,-.5,.5)
> y <- runif(1000,-.5,.5)
> scuad <- x^2+y^2
> lscuad <- scuad < .5^2 & scuad > .4^2
> # Selecciona los valores que conforman el anillo delimitado
> #por 0.4 y 0.5
> Anillo <- data.frame(x[lscuad],y[lscuad])
> plot(Anillo, asp=1)
>
> RipsFiltration <- ripsDiag(Anillo, maxdimension = 1, maxscale = 0.4,
+                            dist = "euclidean", printProgress = TRUE)
# Generated complex of size: 2573249
# Persistence timer: Elapsed time [ 0.000000 ] seconds
> plot(RipsFiltration[['diagram']])
```

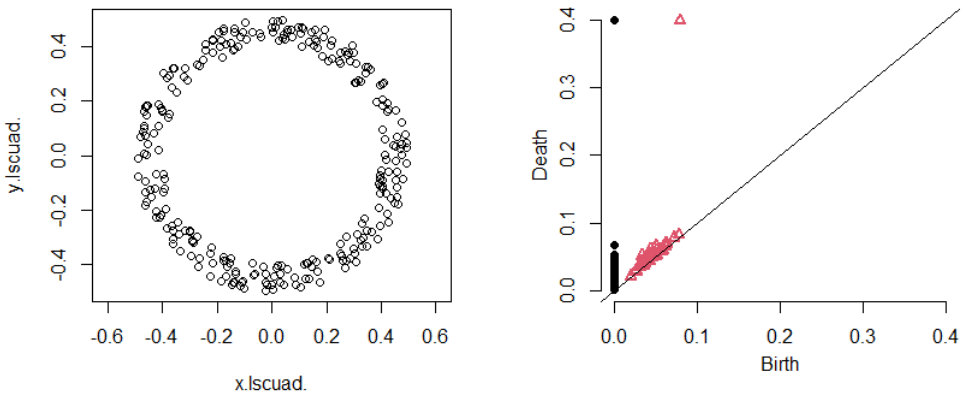


**Figura 2.7.** Nube de puntos a la izquierda y diagrama de persistencia del anillo a la derecha

Observamos que hay una componente conexa y un ciclo que persisten, pues el anillo conforma una única componente conexa y tiene un agujero.

Si calculáramos el diagrama con una filtración basada en el complejo alpha

```
> AlphaFiltration <- alphaComplexDiag(Anillo, maxdimension = 1,
+                                     printProgress = TRUE)
# Generated complex of size: 1621
# Persistence timer: Elapsed time [ 0.000000 ] seconds
```

obtenemos muchos menos símplices en la filtración (1621) que en el caso de Rips (2573249). A pesar de que en este ejemplo ambos computan la filtración de for-

ma rápida, si partiéramos de una nube de puntos considerablemente mayor, el tiempo de ejecución variaría considerablemente, pues como podemos observar, las filtraciones de Rips son menos eficientes que las alfa, ya que las de Rips calculan una mayor cantidad de símplices.

Por todas estas razones computacionales, utilizaremos la filtración alfa en lugar de la de Rips para los ejemplos en los que tengamos una nube de puntos muy grande.

*Ejemplo 2.4 (Dos circunferencias).*

```
> # Generar puntos aleatorios en dos circunferencias
> Circle1 <- circleUnif(n = 100, r = 0.75)
> Circle2 <- circleUnif(n = 100, r = 1) + 2
> Circles <- rbind(Circle1, Circle2)
>
> plot(Circles, asp=1)
>
> # Diagrama de persistencia
> RipsFiltration <- ripsDiag(Circles, maxdimension = 1,
+                            maxscale = 1, dist = "euclidean",
+                            printProgress = TRUE)
# Generated complex of size: 43374
# Persistence timer: Elapsed time [ 0.000000 ] seconds
> plot(RipsFiltration[['diagram']])
```

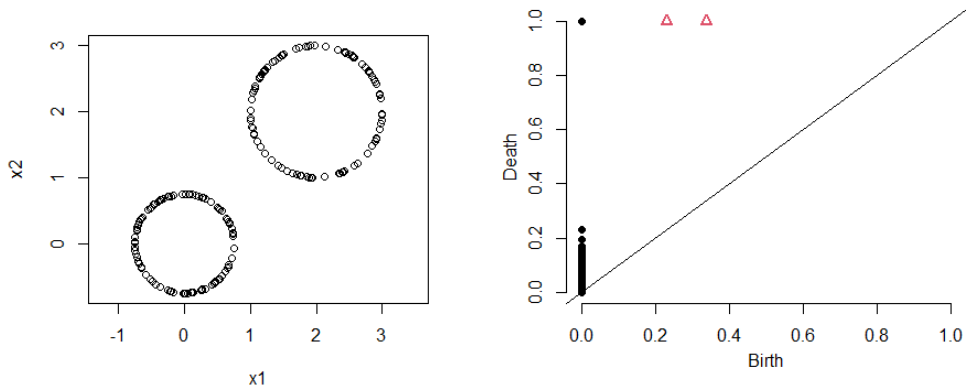


**Figura 2.8.** Nube de puntos a la izquierda y el diagrama de persistencia de las dos circunferencias a la derecha

Se puede observar claramente la persistencia de dos ciclos (los agujeros de cada circunferencia). En cuanto a las componentes conexas, se aprecia la persistencia notable de una de ellas, y otra con una persistencia significativa (compárese dicho diagrama con la figura 2.7 en el que todos los puntos referidos a las componentes estaban completamente juntos).

*Ejemplo 2.5 (Esfera).*

```
> library(rgl) # Librería para hacer plots en 3D
> # Esfera de radio 1 y dimensión 2 generada con 200 puntos
> sphereSample <- sphereUnif(n = 200, d = 2, r = 1)
> plot3d(sphereSample) # Visualizar la nube de puntos
>
> # Diagrama de persistencia alfa
> DiagAlphaCmplx <- alphaComplexDiag(sphereSample,
+                                    library = c("GUDHI", "Dionysus"),
+                                    location = FALSE,
+                                    printProgress = FALSE)
> plot(DiagAlphaCmplx[["diagram"]])
```

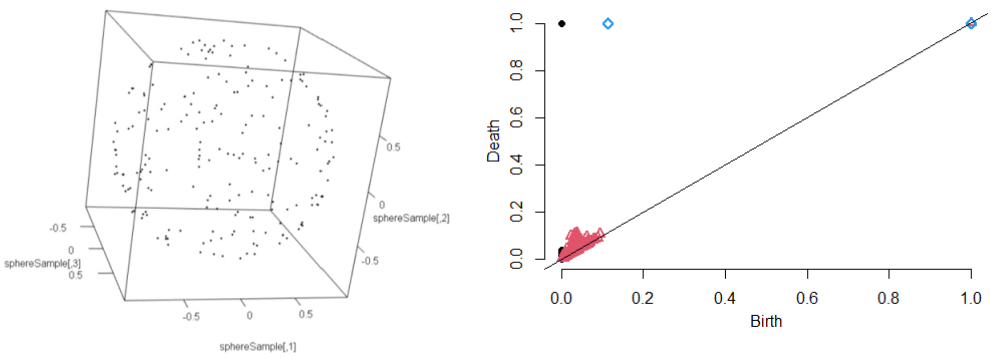


**Figura 2.9.** Nube de puntos a la izquierda y diagrama de persistencia de la esfera a la derecha

Persisten una componente conexa y un hueco (interior de la esfera).

*Ejemplo 2.6 (Toro).*

```
> # Toro generado con 800 puntos con radio del tubo 1.8
> # y radio 5 del agujero central
> torusSample <- torusUnif(n = 800, a = 1.8, c = 5)
> plot3d(torusSample) # Visualizar la nube de puntos
>
```

```
> # Diagrama de persistencia alfa
> DiagAlphaCmplx <- alphaComplexDiag(
+   torusSample, library = c("GUDHI", "Dionysus"),
+   location = FALSE, printProgress = FALSE)
>
> plot(DiagAlphaCmplx[["diagram"]])
```

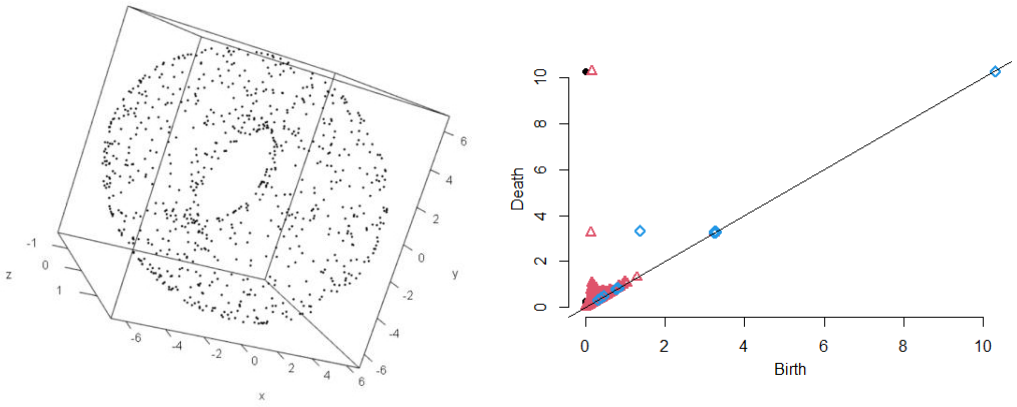


**Figura 2.10.** Nube de puntos a la izquierda y diagrama de persistencia del toro a la derecha

Observamos una componente conexa y un ciclo de dimensión uno (una de las circunferencias generadoras del toro) que persisten hasta el final de la filtración. También se puede apreciar un hueco (hueco característico del toro) y un ciclo de dimensión uno (otra de las circunferencias generadoras) con una persistencia menor pero significativa.
El ciclo con mayor persistencia corresponde a la circunferencia que genera el agujero central, dado que su radio es 5, considerablemente mayor que el radio de la circunferencia que representa el tubo del toro (1.8). Debido a que esta última circunferencia tiene un radio más pequeño, los puntos se conectarán más rápidamente entre sí en comparación con la otra circunferencia, obteniendo así una persistencia menor.

# 3

# Estudio con datos reales

El objetivo de este último capítulo es utilizar la homología persistente para realizar un estudio de los datos genéticos de una especie de estrella de mar llamadada *Coscinasterias tenuispina* (ver figura 3.1). Este es un equinodermo (identificado por Lamarck en 1816) que está ampliamente distribuido a ambos lados del Océano Atlántico y por todo el Mar Mediterráneo. Estas estrellas tienen sistemas reproductivos complejos que combinan ciclos y/o etapas sexuales y asexuales. En la reproducción asexual, los adultos de esta especie pueden dividirse en dos nuevos individuos vivos con una gran capacidad de colonización. Por otro lado, cuando *Coscinasterias tenuispina* se reproduce sexualmente, libera una larva que puede vivir y dispersarse en corrientes de agua durante varias semanas. Esta capacidad de dispersión y colonización ha motivado el estudio de su estructura genética poblacional.

En nuestro caso particular, estudiaremos las relaciones genéticas entre 16 poblaciones distintas del Atlántico y Mediterráneo con un total de 405 individuos basándonos en los datos genéticos utilizados en [12]. Las muestras se han tomado en distintos años (la mayoría, indicados al lado del nombre de cada población), concretamente entre los años 2010 y 2014.

Así, en primer lugar abordaremos algunos conceptos fundamentales sobre genética, ya que serán esenciales para comprender el formato de nuestros datos y las medidas a utilizar para evaluar las relaciones entre las muestras. Posteriormente utilizaremos el paquete TDA previamente explicado para computar la homología persistente, para finalmente intentar extraer información relevante de los resultados obtenidos.

**Figura 3.1.** Imagen de un individuo de *C. tenuispina* de Tenerife obtenida de Gobierno de Canarias

## 3.1. Nociones básicas de genética

La **genética** es una rama de la biología que estudia la herencia y la variación de los caracteres y rasgos en los seres vivos. Se centra en el análisis de los **genes**, los cuales son secuencias de ADN que contienen información que determina las características y el funcionamiento de los organismos. Cada gen se encuentra en una ubicación específica en los cromosomas, conocida como **locus**. Aunque en general, el término 'locus' se extiende a cualquier lugar del genoma de un organismo.
Las posibles variaciones de un locus genético se denominan **alelos**. En nuestro caso trabajaremos con individuos **diploides**, es decir, individuos que tienen dos copias de cada cromosoma en su genoma. En este contexto, en un locus específico, se clasifica a los individuos como **heterocigóticos** u **homocigóticos** según posean dos alelos diferentes en el locus o no, respectivamente.

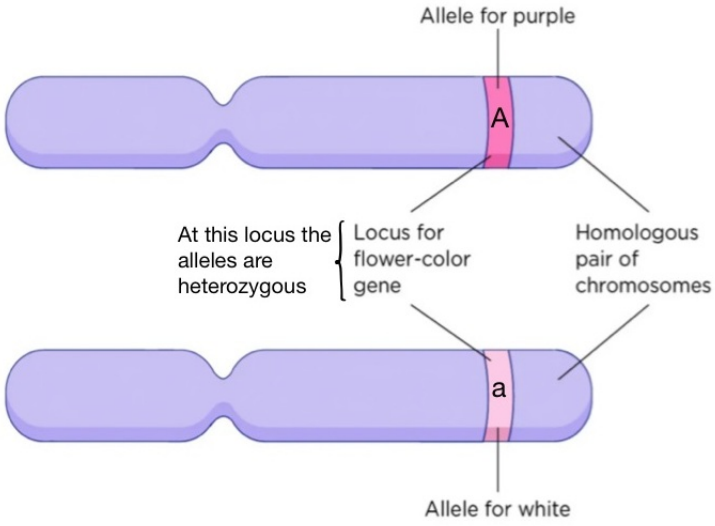


**Figura 3.2.** Ejemplo de locus y alelos

Por otro lado, se denomina **genotipo** a la combinación de alelos que posee un individuo en un locus específico. Por ejemplo, en la figura 3.2, dicho individuo posee el genotipo Aa en el locus del gen que determina el color de la flor.

En el campo de la genética, los **microsatélites** se consideran elementos de gran relevancia para el estudio de las características genéticas en diversos organismos. Estos son secuencias cortas de ADN en las que un fragmento se repite de manera consecutiva. La variación en el número de repeticiones crea diferentes alelos. Estas repeticiones nucleotídicas altamente variables permiten analizar la diversidad genética, lo que los convierte en una herramienta esencial para identificar y caracterizar la variabilidad genética dentro de una población o especie, así como para estudiar la relación de parentesco entre individuos.


**Figura 3.3.** Ejemplo de microsatélite

## 3.2. Genética de poblaciones

La **genética de poblaciones** es una disciplina que se dedica al estudio de la estructura genética de una especie y de las fuerzas que causan cambios en su composición.

Dentro de este ámbito, se define una **población** como un conjunto de individuos de la misma especie que coexisten en un mismo tiempo y espacio, lo cual les permite cruzarse e intercambiar genes.
Existen varios métodos para evaluar la diversidad genética en diferentes niveles de una población. Uno de ellos es el coeficiente $F_{ST}$, el cual es una medida utilizada para determinar la diferenciación genética entre distintas subpoblaciones de una población total. A menor valor de $F_{ST}$, mayor proximidad entre dichas poblaciones.
Dadas dos poblaciones X e Y de las que se posee la información alélica en $r$ locus distintos, el coeficiente $F_{ST}$ (de Latter, 1972) entre dichas poblaciones se calcula como sigue ([13])

$$F_{ST} = \frac{(J_X + J_Y)/2 - J_{XY}}{1 - J_{XY}}$$

donde

$$J_X = \sum_{j}^{r} \sum_{i}^{m_j} \frac{x_{ij}^2}{r}, \quad J_Y = \sum_{j}^{r} \sum_{i}^{m_j} \frac{y_{ij}^2}{r}, \quad J_{XY} = \sum_{j}^{r} \sum_{i}^{m_j} \frac{x_{ij} y_{ij}}{r}$$

con $x_{ij}$ e $y_{ij}$ las frecuencias del i-ésimo alelo del j-ésimo locus en las poblaciones X e Y, respectivamente. Es decir, $x_{ij}$ sería el cociente del número de copias presentes del alelo i-ésimo en el j-ésimo locus y el número total de alelos en dicho locus de todos los individuos de la población X, e igual con $y_{ij}$ pero considerando la población Y. Por otro lado, $m_j$ es el número de alelos del j-ésimo locus.

*Ejemplo 3.1.* En Datos poblaciones ESC y ALI (de [21]) se recoge la información genética de dos poblaciones distintas de *C. tenuispina*, ESC (Los Escullos, Almería) y ALI (Alicante, Murcia). Concretamente, se muestran los alelos correspondientes a 12 microsatélites (dos alelos por cada uno) de 32 individuos (20 de ESC y 12 de ALI). Veamos cuál es el coeficiente $F_{ST}$ asociado.
En primer lugar calculamos las frecuencias de cada alelo en su respectivo locus (microsatélite) de cada población, las cuales se muestran en la siguiente figura 3.4

| | ESC | |
|---|---|---|
| MICROSATÉLITE | ALELOS | xij |
| 1FAM | 171 | 1 |
| | | |
| 6NED | 160 | 0,5 |
| | 163 | 0,5 |
| 14HEX | 138 | 0,125 |
| | 137 | 0,875 |
| 31FAM | 297 | 1 |
| | | |
| 30NED | 397 | 0,5 |
| | 405 | 0,5 |
| 32FAM | 245 | 0,5 |
| | 249 | 0,5 |
| 19HEX | 133 | 0,5 |
| | 150 | 0,5 |
| 40FAM | 152 | 1 |
| | | |
| 24HEX | 365 | 1 |
| | | |
| 25HEX | 294 | 0,5 |
| | 297 | 0,5 |
| 27FAM | 293 | 0,95 |
| | 295 | 0,05 |
| 13FAM | 359 | 1 |
| | | |

| | ALI | |
|---|---|---|
| MICROSATÉLITE | ALELOS | yij |
| 1FAM | 171 | 1 |
| | | |
| 6NED | 160 | 0,5 |
| | 163 | 0,5 |
| 14HEX | 138 | 0,5 |
| | 137 | 0,5 |
| 31FAM | 297 | 1 |
| | | |
| 30NED | 397 | 0,5 |
| | 405 | 0,5 |
| 32FAM | 245 | 0,5 |
| | 249 | 0,5 |
| 19HEX | 133 | 1 |
| | 150 | 0 |
| 40FAM | 152 | 1 |
| | | |
| 24HEX | 365 | 1 |
| | | |
| 25HEX | 294 | 0 |
| | 297 | 1 |
| 27FAM | 293 | 1 |
| | 295 | 0 |
| 13FAM | 359 | 1 |
| | | |

**Figura 3.4.** Frecuencias alélicas de las poblaciones ESC (derecha) y ALI (izquierda) para cada microsatélite

Nótese que la población ESC, al tener 20 individuos, se tienen 40 alelos por cada microsatélite en total (pues tenemos dos alelos en un microsatélite por cada individuo), mientras que en ALI, al tener 12 individuos hay 24 alelos en total. Así, por ejemplo para calcular la frecuencia relativa del alelo 160 del microsatélite 6NED, bastaría con contar cuántos alelos 160 hay en dicho microsatélite y dividirlo entre 40 y 12 para las poblaciones de ESC y ALI, respectivamente.
Con estos valores y teniendo en cuenta que $r = 12$ (pues se han muestreado 12 microsatélites en cada población), obtenemos

$$J_X \approx 0.7655,\ J_Y \approx 0.8333,\ J_{XY} \approx 0.7458$$

Así, podemos calcular finalmente el coeficiente $F_{ST}$ como

$$F_{ST} = \frac{(J_X + J_Y)/2 - J_{XY}}{1 - J_{XY}} \approx 0.2109$$

Dado que el coeficiente $F_{ST}$ nos indica cuán próximas se encuentran determinadas poblaciones en el contexto genético, nos referiremos a él como 'distancia genética'.

## 3.3. Estudio de los datos

Una vez introducidas todas las nociones anteriores, estamos en condiciones de tratar con los datos que se mencionaron al principio del capítulo.
Los datos vienen dados en una tabla de hoja de cálculo obtenida de [12] en la que se recoge información genética de individuos de 16 poblaciones distintas (o de una misma población pero en distintos años). Dicha tabla se puede consultar en Datos C. Tenuispina de la carpeta [21]. La información que muestra se estructura como sigue:

- Primera columna: individuos
- Segunda columna: poblaciones
- Tercera columna en adelante: los distintos microsatélites y sus correspondientes alelos (2 alelos por cada microsatélite) en cada individuo

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 12 | 507 | 24 | 23 | 20 | 24 | 12 | 24 | 26 |
| 2 | Title line: "Coscinasterias_microsatellites_total" | | | HER_2014 | ESC_2014 | ALI_2010 | ALI_2014 | CUN_2012 | PAL_2011 |
| 3 | Sample | Pop | 1FAM | | 6NED | | 14HEX | | 31FAM |
| 4 | HER_2014_01 | HER_2014 | 171 | 177 | 160 | 160 | 137 | 137 | 297 |
| 5 | HER_2014_02 | HER_2014 | 171 | 177 | 160 | 160 | 137 | 137 | 297 |
| 6 | HER_2014_03 | HER_2014 | 171 | 177 | 160 | 160 | 137 | 138 | 297 |
| 7 | HER_2014_04 | HER_2014 | 171 | 171 | 160 | 160 | 137 | 138 | 297 |
| 8 | HER_2014_05 | HER_2014 | 171 | 177 | 160 | 160 | 137 | 138 | 297 |
| 9 | HER_2014_06 | HER_2014 | 171 | 177 | 160 | 160 | 137 | 138 | 297 |
| 10 | HER_2014_07 | HER_2014 | 171 | 177 | 160 | 160 | 137 | 138 | 297 |
| 11 | HER_2014_08 | HER_2014 | 171 | 177 | 160 | 160 | 137 | 138 | 297 |
| 12 | HER_2014_09 | HER_2014 | 171 | 171 | 160 | 160 | 137 | 137 | 297 |
| 13 | HER_2014_10 | HER_2014 | 171 | 177 | 160 | 160 | 137 | 137 | 297 |
| 14 | HER_2014_11 | HER_2014 | 171 | 177 | 160 | 160 | 137 | 137 | 297 |
| 15 | HER_2014_12 | HER_2014 | 171 | 177 | 160 | 160 | 137 | 138 | 297 |
| 16 | HER_2014_13 | HER_2014 | 171 | 171 | 160 | 160 | 137 | 137 | 297 |
| 17 | HER_2014_14 | HER_2014 | 171 | 177 | 160 | 160 | 137 | 137 | 297 |
| 18 | HER_2014_15 | HER_2014 | 171 | 177 | 160 | 160 | 137 | 137 | 297 |
| 19 | HER_2014_16 | HER_2014 | 171 | 177 | 160 | 160 | 137 | 137 | 297 |
| 20 | HER_2014_17 | HER_2014 | 171 | 177 | 160 | 163 | 137 | 137 | 297 |
| 21 | HER_2014_18 | HER_2014 | 171 | 177 | 163 | 163 | 137 | 137 | 297 |
| 22 | HER_2014_19 | HER_2014 | 171 | 177 | 160 | 160 | 137 | 137 | 297 |
| 23 | HER_2014_20 | HER_2014 | 171 | 177 | 160 | 160 | 137 | 137 | 297 |
| 24 | HER_2014_21 | HER_2014 | 171 | 177 | 160 | 160 | 137 | 137 | 297 |
| 25 | HER_2014_22 | HER_2014 | 171 | 177 | 160 | 160 | 137 | 137 | 297 |
| 26 | HER_2014_23 | HER_2014 | 171 | 177 | 160 | 160 | 137 | 137 | 297 |
| 27 | ESC_2014_01 | ESC_2014 | 171 | 171 | 160 | 163 | 138 | 138 | 297 |
| 28 | ESC_2014_02 | ESC_2014 | 171 | 171 | 160 | 163 | 137 | 138 | 297 |
| 29 | ESC_2014_03 | ESC_2014 | 171 | 171 | 160 | 163 | 138 | 138 | 297 |
| 30 | ESC_2014_04 | ESC_2014 | 171 | 171 | 160 | 163 | 138 | 138 | 297 |
| 31 | ESC_2014_05 | ESC_2014 | 171 | 171 | 160 | 163 | 138 | 138 | 297 |

**Figura 3.5.** Datos (parciales) de *Coscinasterias ternuispina*

Cada población está etiquetada por abreviaturas, las cuales se muestran en la tabla 3.1.

**Tabla 3.1.** Poblaciones junto con sus etiquetas

| Etiqueta | Localización |
|---|---|
| ABA | Abades, Tenerife, España |
| ALI | Alicante, Murcia, España |
| BOCA | Bocacangrejo, Tenerife, España |
| CAI | Cadaqués, Girona, España |
| CUN | Cunit, Tarragona, España |
| ESC | Los Escullos, Almería, España |
| GIJ | Gijón, Asturias, España |
| HER | La Herradura, Granada, España |
| KNI | Cnidos, Datça, Turquía |
| LAN | Playa Blanca, Lanzarote, España |
| LLA | Llançà, Girona, España |
| NAP | Nápoles, Campania, Italia |
| PAL | Pálamos, Girona, España |
| PK | Plakias, Creta, Grecia |
| SIC | Taormina, Sicilia, Italia |
| TAZ | Tazacorte, La Palma, España |

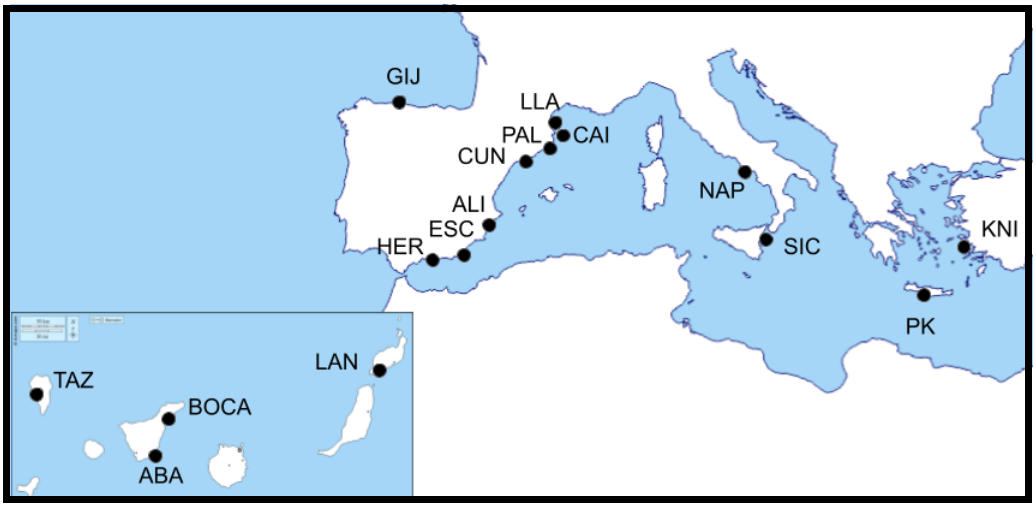


**Figura 3.6.** Poblaciones de *Coscinasterias ternuispina* del estudio

Para analizar posibles relaciones genéticas entre las diferentes poblaciones utilizando herramientas de homología persistente, utilizaremos el paquete `hierftstat` de R el cual nos proporciona, entre otras ([9]), la función `genet.dist`, que nos permite calcular la distancia genética entre poblaciones mediante el coeficiente $F_{ST}$. Sus argumentos son los siguientes, con indicación de los valores por defecto cuando los hay:

```
genet.dist(dat, diploid=TRUE, method="Dch")
```

El argumento `dat` debe ser un data frame que contenga las poblaciones en la primera columna y en las siguientes sus respectivos genotipos en los distintos locus que se han muestreado. Por otro lado, se fija `diploid=TRUE` cuando estamos considerando individuos diploides. Finalmente con `method` indicamos el método con el que queramos calcular la distancia genética. La función proporciona una gran cantidad de métodos a considerar, sin embargo nosotros nos limitaremos a `method="Fst"`.
Como salida, se obtiene una matriz de distancias entre cada par de poblaciones.

Para utilizar dicha función con nuestros datos, en primer lugar debemos importar la tabla excel a R

```
> library(hierfstat)
> library(TDA)
> library(readxl) # Paquete para cargar tabla excel
> # Para importar tabla excel:
> my_data <- data.frame(read_xlsx(
+             "C:RUTA DE LOS DATOS/nombre_tabla.xlsx", skip=2))
```

De esta forma, obtenemos un data frame con la información de la tabla excel (utilizamos `skip=2` para no considerar las dos primeras filas de la tabla, las cuales no nos estaban dando información esencial de los datos).
En dicha tabla resulta que hay casillas con un valor de 0, lo cual indicaba la falta de información sobre los alelos del microsatélite específico para ese individuo. En R indicamos esta falta de datos con el valor `NA`.

```
> my_data[my_data==0] <- NA       # Donde haya 0 ponemos NA
```

Para obtener los genotipos de cada microsatélite ejecutamos el siguiente código:

```
> # Todas las filas y las columnas con paso 2
> ct <- my_data[,c(1,2,1+2*(1:12))]
> # Pegamos los alelos que faltan
> for (i in 1:12){
+   ct[,i+2] <- as.numeric(paste(my_data[,1+2*i],my_data[,2+2*i],
+                                sep=''))
+ }
```

Así, en `ct` tenemos almacenados los genotipos asociados a cada microsatélite, una muestra de la estructura de este data frame es la siguiente

```
> ct[1:10, 1:7]
        Sample      Pop  X1FAM  X6NED X14HEX X31FAM X30NED
1  HER_2014_01 HER_2014 171177 160160 137137 297297 405405
2  HER_2014_02 HER_2014 171177 160160 137137 297297 405405
3  HER_2014_03 HER_2014 171177 160160 137138 297297 405405
4  HER_2014_04 HER_2014 171171 160160 137138 297297 405405
```

```
5  HER_2014_05 HER_2014 171177 160160 137138 297297 405405
6  HER_2014_06 HER_2014 171177 160160 137138 297297 405405
7  HER_2014_07 HER_2014 171177 160160 137138 297297 405405
8  HER_2014_08 HER_2014 171177 160160 137138 297297 405405
9  HER_2014_09 HER_2014 171171 160160 137137 297297 405405
10 HER_2014_10 HER_2014 171177 160160 137137 297297 405405
```

A continuación usamos la función `genet.dist` para obtener la matriz de distancias y calculamos su correspondiente diagrama de persistencia

```
> # Matriz de distancias genéticas:
> Mdist <- genet.dist(ct[,-1], diploid=TRUE, method="Fst")
>
>
> # Complejo de Rips:
> DiagRips <- ripsDiag(X = Mdist, maxdimension = 2,
+                      maxscale = 0.6, dist='arbitrary',
+                      library = c("Dionysus"), location = TRUE,
+                      printProgress = FALSE)
>
> # Visualizar el diagrama de persistencia
> plot(DiagRips$diagram)
```

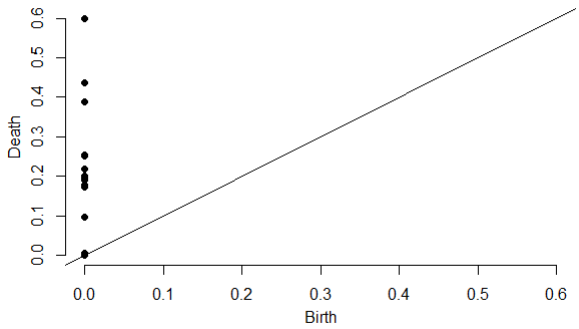


**Figura 3.7.** Diagragma de persistencia de las poblaciones de individuos *C. Tenuispina*

Nótese que en este caso, los vértices de nuestra filtración se corresponden con las distintas poblaciones. Luego cada población se conectará con las demás en un orden u otro en función de las distancia genética entre ellas, hasta finalmente formar una única componente conexa. Además, en este caso el índice de persistencia nos indicará cuán parecidas serán dichas poblaciones.

A mayor índice de persistencia, menos parentesco tendrán las poblaciones que se vayan uniendo.

Más específicamente, obtenemos la siguiente información

```
> DiagRips$diagram
      dimension Birth       Death
 [1,]         0     0 0.600000000
 [2,]         0     0 0.250393833
 [3,]         0     0 0.001893939
 [4,]         0     0 0.201632220
 [5,]         0     0 0.095515500
 [6,]         0     0 0.189483484
 [7,]         0     0 0.389266378
 [8,]         0     0 0.253462054
 [9,]         0     0 0.436551798
[10,]         0     0 0.172958288
[11,]         0     0 0.001904762
[12,]         0     0 0.001219512
[13,]         0     0 0.005996003
[14,]         0     0 0.192305346
[15,]         0     0 0.177595697
[16,]         0     0 0.219403674
```

Para ser capaces de interpretar esta información, debemos identificar qué poblaciones son las que se han unido en los distintos valores de la filtración, esto lo obtendremos a través del elemento `cycleLocation` (como vimos en la sección anterior), además, los distintos valores que aparecerán en `cycleLocation` se corresponderán con el orden de las poblaciones en la matriz de distancias.

```
> # Etiquetas matriz de distancias
> attr(Mdist, "Labels")
 [1] "ABA"      "ALI_2010" "ALI_2014" "BOCA"     "CAI_2011"
 "CUN_2012" "ESC_2014" "GIJ"      "HER_2014"
[10] "KNI"      "LAN"      "LLA_2011" "LLA_2014" "NAP_2013"
"PAL_2011" "PAL_2014" "PK"       "SIC"
[19] "TAZ"
```

Así, el valor `1` que aparezca en `cycleLocation` se corresponderá con la población ABA, el `2` con ALI_2010, y así sucesivamente.

A continuación, estudiaremos las componentes conexas en torno a los distintos valores que parece que se acumulan en el diagrama de persistencia.

En primer lugar, identificamos las componentes conexas que aparecen muy cercanas a la diagonal

```
> # --------------- cercanos a 0 --------------
>
> # Obtenemos los índices de las filas que cumplen
> # dicha condición
> filas <- which(DiagRips$diagram[,3]<0.05)
>
> # Ver las poblaciones que se conectan (de 2 en 2)
> Pob <- c()
> for (i in filas){
+ for (j in 1:2){
+ Pob <- c(Pob, attr(Mdist, "Labels")[DiagRips$cycleLocation[[i]]
[j]])}
+ Pob <- c(Pob, DiagRips$diagram[[i,3]]) # Valor en el que muere
+ Pob <- c(Pob, '')} # Para dejar un espacio
> Pob
 [1] "ALI_2014"        "ALI_2010"        "0.00189393939393896" ""
 [5] "LLA_2011"        "ALI_2014"        "0.00190476190476169" ""
 [9] "LLA_2014"        "ALI_2014"        "0.00121951219512214" ""
[13] "NAP_2013"        "ALI_2014"        "0.00599600266489018" ""
```

Así, parece que las poblaciones ALI, NAP Y LLA, están muy próximas genéticamente, conformando una única componente conexa (que acaba en Nápoles) con un índice de persistencia muy bajo. En particular, las poblaciones más próximas genéticamente son LLA_2014 y ALI_2014.
En la Discusión de [12] se comprueba la existencia de un *superclon* (un linaje clonal dominante) que está ampliamente distribuido por las costas del Mediterráneo occidental de la mayoría de poblaciones estudiadas de la zona, desde Alicante hasta Nápoles. Nuestros resultados parecen confirmar este hecho, pues obtenemos distancias genéticas muy pequeñas entre las poblaciones mencionadas. Nótese que de las poblaciones PAL y CUN no obtenemos ninguna información, pues estas, junto con ALI_2014 poseen una distancia genética exactamente igual a cero (es decir, son poblaciones clónicas), luego es probable que esta técnica detecte que se tratan de exactamente el mismo vértice y por ello no se muestran conexiones entre ellas. Sin embargo, implícitamente sí están conectadas luego podemos efectivamente justificar esa presencia del *superclon* con los resultados obtenidos.
Además, en [12] también se expone que ese *superclon* no está presente en las zonas relativas al Mediterráneo oriental, Océano Atlántico y el mar de Alborán. Esto lo podemos intuir puesto que las siguientes conexiones entre poblaciones que obtenemos (cuando estudiamos los puntos en torno a 0.2) poseen una distancia

genética considerablemente mayor, y gran parte de las conexiones significativas, se producen entre poblaciones distintas a las mencionadas con anterioridad.

A continuación, estudiemos las componentes conexas en torno a 0.1

```
> # --------------- Puntos aislados por 0.1 ----------------
>
> # Obtenemos los índices de las filas que cumplen
> # dicha condición
> filas <- which(DiagRips$diagram[,3]<0.1 &
+                DiagRips$diagram[,3]>0.05)
>
> # Ver las poblaciones que se conectan (de 2 en 2)
> Pob <- c()
> for (i in filas){
+ for (j in 1:2){
+ Pob <- c(Pob, attr(Mdist, "Labels")[DiagRips$cycleLocation[[i]]
[j]])}
+ Pob <- c(Pob, DiagRips$diagram[[i,3]]) # Valor en el que muere
+ Pob <- c(Pob, '')} # Para dejar un espacio
> Pob
[1] "CAI_2011"       "ALI_2014"       "0.0955155003859012" ""
>
```

En este caso, obtenemos una única componente conexa formada por la unión de CAI_2011 y ALI_2014. Ampliamos así la componente formada anteriormente, resultando una significativa proximidad genética entre las poblaciones de ALI, NAP, LLA y CAI.

En el rango en torno a 0.2 obtenemos lo que sigue

```
> # ------------- Puntos en torno a 0.2 -----------
> # Obtenemos los índices de las filas que cumplen
> # dicha condición
> filas <- which(DiagRips$diagram[,3]<0.3 &
+                DiagRips$diagram[,3]>0.1)
>
> # Ver las poblaciones que se conectan (de 2 en 2)
> Pob <- c()
> for (i in filas){
+ for (j in 1:2){
+ Pob <- c(Pob, attr(Mdist, "Labels")[DiagRips$cycleLocation[[i]]
[j]])}
+ Pob <- c(Pob, DiagRips$diagram[[i,3]]) # Valor en el que muere
```

```
+ Pob <- c(Pob, '')} # Para dejar un espacio
> Pob
 [1] "ALI_2010"      "ABA"           "0.250393832756963" ""
 [5] "BOCA"          "ABA"           "0.201632220391638" ""
 [9] "ESC_2014"      "ALI_2014"      "0.189483484455959" ""
[13] "HER_2014"      "BOCA"          "0.253462053590231" ""
[17] "LAN"           "BOCA"          "0.172958287516577" ""
[21] "PK"            "LAN"           "0.192305345683326" ""
[25] "SIC"           "LAN"           "0.177595697441764" ""
[29] "TAZ"           "ABA"           "0.219403673874873" ""
```

A partir de este rango, comenzamos a obtener poblaciones que no están tan relacionadas genéticamente entre sí como las anteriores, ya que obtenemos valores de $F_{ST}$ superiores.
Resulta que LAN (Lanzarote) y BOCA (Tenerife) son los más próximos genéticamente en este caso. Podría esperarse que las poblaciones correspondientes a las Islas Canarias tuvieran más proximidad genética entre sí que con las poblaciones del Mediterráneo, sin embargo, obtenemos que SIC (Italia) y LAN (Lanzarote) poseen menos diferenciación genética que TAZ (La Palma) y ABA (Tenerife). Además, resulta también que PK (Grecia) y LAN son de las poblaciones más próximas en este rango, lo que nos podría indicar una cierta dispersión de las larvas entre Lanzarote y poblaciones geográficamente muy alejadas del Mediterráneo. Este hecho confirma la gran capacidad de dispersión que pueden llegar a tener las larvas a través de las corrientes marinas como se exponía en [12]. Y que además, la proximidad geográfica no tendría por qué implicar proximidad genética, puesto que como hemos comprobado, esta especie tiene una gran capacidad de dispersión y podría darse que las larvas acaben en puntos geográficamente muy alejados respecto de la población de origen.

Finalmente, mostramos las componentes con valores de la filtración cercanos a 0.4

```
> # ------------ Puntos en torno a 0.4 -------------
>
> # Obtenemos los índices de las filas que cumplen
> # dicha condición
> filas <- which(DiagRips$diagram[,3]<0.5 &
+                DiagRips$diagram[,3]>0.3)
>
> # Ver las poblaciones que se conectan (de 2 en 2)
> Pob <- c()
> for (i in filas){
+ for (j in 1:2){
```

```
+ Pob <- c(Pob, attr(Mdist, "Labels")[DiagRips$cycleLocation[[i]]
[j]])}
+ Pob <- c(Pob, DiagRips$diagram[[i,3]]) # Valor en el que muere
+ Pob <- c(Pob, '')} # Para dejar un espacio
> Pob
[1] "GIJ"            "ABA"            "0.389266378421042" ""
[6] "KNI"            "BOCA"           "0.436551798078657" ""
```

Estos últimos valores nos indican las poblaciones más distantes genéticamente. Aparecen tanto GIJ como KNI, las cuales no se habían observado en los casos anteriores, esto podría indicar que dichas poblaciones se encuentran genéticamente aisladas del resto, ya que las distancias genéticas que obtenemos son muy grandes, indicando una significativa diferenciación genética.

La información genética entre las distintas poblaciones que hemos obtenido viene recogida de forma resumida en la siguiente figura

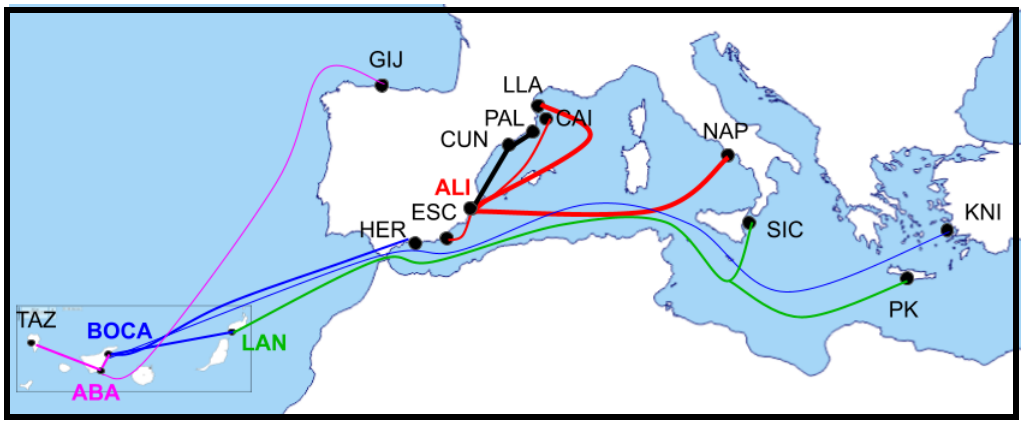


**Figura 3.8.** Componentes conexas que obtuvimos entre poblaciones de *C. tenuispina* (en los casos de los que disponíamos de dos muestras de una misma población, en esta figura se representa la muestra más reciente). A mayor grosor de la línea, mayor relación genética (y por tanto, menor valor en la filtración). El color de cada línea nos indica la población de destino en cada conexión (a excepción de las negras, que señalan las poblaciones clónicas), por ejemplo la línea que empieza en GIJ se une a ABA (color rosa) y se corresponde con la conexión GIJ-ABA que obtenemos en la filtración con un valor de 0.3829

## 3.4. Conclusiones

El objetivo de este trabajo consistía en introducir los fundamentos teóricos de la homología persistente y el uso de herramientas computacionales para su aplicación. Así, hemos realizado un amplio recorrido desde nociones básicas como los complejos simpliciales hasta la homología persistente, tanto desde el punto de vista teórico como computacional. En este último capítulo, se propuso estudiar una gran cantidad datos genéticos sobre una especie de estrella de mar

llamada *Coscinasterias tenuispina* para intentar relacionar los resultados obtenidos con algunos de los expuestos en [12], conseguidos mediante un tratamiento de datos esencialmente estadístico.

Así, en esta memoria hemos analizados las características homológicas en dimensión cero (componentes conexas) de una filtración de Rips tomando como vértices distintas poblaciones de *C. tenuispina*. Hay que tener en cuenta que para un conjunto de vértices tan pequeño (19 en total, considerando también las que aparecían muestreadas en distintos años) es difícil obtener información relevante en dimensiones superiores. Con un mayor conjunto de datos, tal vez podríamos haber obtenido ciclos de dimensión uno o incluso de mayor dimensión. Sin embargo, las componentes conexas de por sí ya nos dan información valiosa respecto a la estructura genética de las poblaciones.
Hemos comprobado mediante la homología persistente esa presencia de un *superclon* que ha colonizado la zona del Mediterráneo Oeste desde Alicante hasta Nápoles.
Por otro lado, hemos confirmado esa gran capacidad de dispersión que pueden llegar a tener las larvas a través de corrientes marinas, encontrando que poblaciones de Lanzarote y Sicilia estaban más próximas genéticamente que Tazacorte con Abades, por ejemplo.

En definitiva, con nuestro estudio hemos sido capaces de hacernos una idea de la estructura genética entre distintas poblaciones de dicha especie, así como comprobar empíricamente su gran capacidad de dispersión y colonización. Sin embargo, este estudio podría ampliarse al análisis de la diversidad genética entre individuos, en vez de poblaciones y estudiar las posibles relaciones entre heterocigosidad y clonalidad, o bien la evolución genética de las poblaciones en el tiempo. Además, podríamos realizar una comparación entre nuestros resultados y los de otro estudio en el que no se consideren los clones de cada población, puesto que es posible que se obtengan resultados significativamente diferentes en ambos casos. También sería importante combinar esta técnica con alguna otra para corroborar las conclusiones que obtenidas. En particular, las propias implementaciones de paquetes de software (como el paquete TDA que hemos usado) suelen incluir funciones para combinar técnicas estadísticas: funciones de densidad, intervalos de confianza, *density clustering*, etc. con los resultados de los algoritmos de ATD.

Esta primera aproximación que hemos realizado está obviamente limitada por cuestiones de tiempo y espacio pero, como vemos, existen infinidad de posibilidades que nos proporciona la homología persistente para dar continuidad al presente trabajo.

# Bibliografía


[1] Björner, A. (1995). *Topological methods. Handbook of combinatorics, 2*, p. 1819–1872. Elsevier Science B. V., Amsterdam, The Netherlands.

[2] Díaz-Sánchez, O. (2019). *Fundamentos de la homología persistente y su código de barras asociado*. Trabajo de Fin de Grado. Grado en Matemáticas. Universidad de La Laguna.

[3] Edelsbrunner, H. (1993, July). The union of balls and its dual shape. In *Proceedings of the ninth annual symposium on Computational geometry* (pp. 218–231).

[4] Edelsbrunner, H., & Harer, J. L. (2010). *Computational topology: an introduction*. American Mathematical Society, Providence, Rhode Island.

[5] Edelsbrunner, H., Kirkpatrick, D., & Seidel, R. (1983). On the shape of a set of points in the plane. *IEEE Transactions on information theory, 29(4)*, 551–559.

[6] Edelsbrunner, H., Letscher, D. & Zomorodian, A. (2002) Topological Persistence and Simplification. *Discrete & Computational Geometry 28(4)* pp. 511–533.

[7] Edelsbrunner, H., & Mücke, E. P. (1994). Three-dimensional alpha shapes. *ACM Transactions On Graphics (TOG), 13(1)*, 43–72.

[8] Fasy, B. T., Kim, J., Lecci, F., Maria, C., Millman, D. L., Rouvreau, V., & Kim, M. J. (2023). TDA: *Statistical Tools for Topological Data Analysis*. R package version 1.9, disponible en https://CRAN.R-project.org/package=TDA.

[9] Goudet, J., Jombart, T., & Goudet, M. J. (2022). hierfstat: *Estimation and Tests of Hierarchical F-Statistics*. R package version 0.5–11, disponible en: https://CRAN.R-project.org/package=hierfstat.

[10] Munkres, J. R. (2018). *Elements of algebraic topology*. CRC press, Boca Raton, Florida.

[11] Otter, N., Porter, M. A., Tillmann, U., Grindrod, P., & Harrington, H. A. (2017). A roadmap for the computation of persistent homology. *EPJ Data Science, 6*, 1–38.

[12] Pérez-Portela, R., Garcia-Cisneros, A., Campos-Canet, M., & Palacín, C. (2021). Genetic homogeneity, lack of larvae recruitment, and clonality in absence of females across western Mediterranean populations of the starfish Coscinasterias tenuispina. *Scientific Reports, 11(1)*, 16819.

[13] Takezaki, N., & Nei, M. (1996). Genetic distances and reconstruction of phylogenetic trees from microsatellite DNA. *Genetics, 144(1)*, 389–399.

[14] Voronoi, G. (1907) Nouvelles applications des paramètres continus á la théorie des formes quadratiques. Premier Mémoire: Sur quelques proriétés des formes quadratiques positive parfaites. *J. Reine Angew. Math, 133*, 97–178.

[15] Voronoi, G. (1908) Nouvelles applications des paramètres continus á la théorie des formes quadratiques. Deuxième Mémoire: Recherches sur les paralléloèdres primitifs. *J. Reine Angew. Math, 134*, 198–287.

[16] Zomorodian, A. & Carlsson, G. (2005) Computing Persistent Homology. *Discrete & Computational Geometry, 33(2)* pp. 249-274.

[17] R Core Team (2023). R: *A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/.

[18] Documentación GUDHI. Disponible en: https://project.inria.fr/gudhi/software/.

[19] Documentación Dionysus. Disponible en: https://www.mrzv.org/software/dionysus/.

[20] Documentación PHAT. Disponible en: https://bitbucket.org/phat-code/phat/src/master/.

[21] Todos los scripts de R así como los datos utilizados a lo largo de la memoria se pueden consultar en la carpeta de drive: Rscripts y datos.

# Persistent homology applied to genetic study of populations

**Belinda Hazel Cáceres Barrera**
Facultad de Ciencias • Sección de Matemáticas
Universidad de La Laguna
alu0101339368@ull.edu.es

## Abstract

*Topological Data Analysis is a discipline that combines concepts and techniques from topology with data analysis to study and understand the underlying structure in complex data sets. Its objective is to reveal important patterns and features that may be hidden in the data and are not easily detectable using traditional analysis methods. One of its widely used tools is persistent homology, which detects relevant topological features through a filtration of simplicial complexes that can be generated from a point cloud. This technique of topological analysis has gained significant relevance in recent years and it is applied in a wide variety of scientific disciplines.*
*In this work, we use the tools provided by persistent homology in the context of genetics to conduct a study among populations of Coscinasterias tenuispina, a species of starfish distributed in various areas of the Atlantic and Mediterranean.*

## 1. Objectives

The main objective of this work is to familiarize ourselves with the theoretical principles of persistent homology and the practical use of computational tools to carry it out, and ultimately, to utilize this acquired knowledge to conduct a genetic study among populations of the starfish ***Coscinasterias tenuispina***, in order to understand their genetic structure and the relationships between them.

**Figure 1:** *Image of Coscinasterias tenuispina from Punta del Hidalgo, Tenerife*

In particular, we studied the populations of *C. tenuispina* across the Atlantic Ocean and Mediterranean Sea.

## 2. Methodology

For this study, we used the tools that provided by persistent homology through a R package called TDA. Despite of the fact that in this memoir we introduce some filtrations which are important in Topological Data Analysis, such as the Alpha and Rips filtrations, we only used the Rips filtration for the starfish study, since we did not use the euclidean distance.

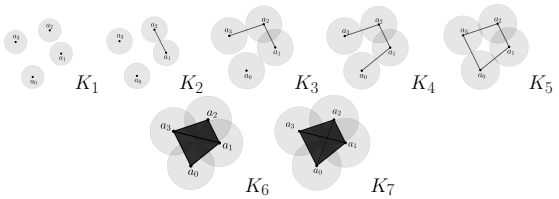


**Figure 2:** *A Rips filtration example*

Through these filtrations we obtain the so-called **persistence diagrams**, which allow us to get relevant information about the topological characteristics of a point cloud.

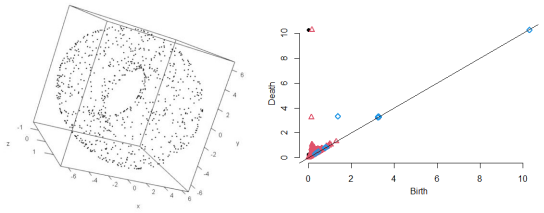


**Figure 3:** *A torus (left) and its persistent diagram (right)*

## 3. Results

We were able to understand the genetic structure of the *C. tenuispina* populations that were studied through persistent homology. Among the results obtained, the most relevant could be that geographic proximity does not necessarily imply genetic proximity, since we found that some populations which were very close geographically did not have a significant genetic relation.

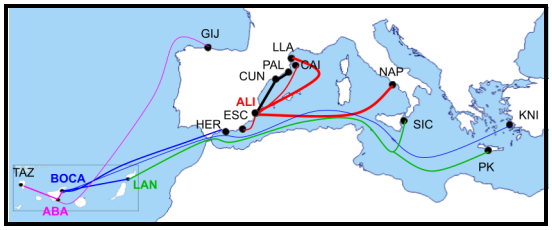


**Figure 4:** *Connected components obtained which indicate the genetic proximity of the populations studied*

## References

[1] Zomorodian, A. & Carlsson, G. (2005) Computing Persistent Homology. *Discrete & Computational Geometry, 33(2)* pp. 249-274.