Centro de Investigación en Matemáticas, A.C.

Análisis Topológico de Datos: Robusticidad y análisis de sensibilidad de algoritmos

T E S I S

Que para obtener el grado de
Maestro en Ciencias
con especialidad en
Probabilidad y Estadística

P r e s e n t a
Lic. Jesús Manuel Pérez Angulo

Director de tesis:
Dr. Víctor Manuel Pérez Abreu Carrión

Guanajuato, Gto. Diciembre de 2016

Centro de Investigación en Matemáticas, A.C.

Acta de Examen de Grado

Acta No.: 116

Libro No.: 002

Foja No.: 116

En la Ciudad de Guanajuato, Gto., siendo las 12:00 horas del día 15 de diciembre del año 2016, se reunieron los miembros del jurado integrado por los señores:

DR. JOSÉ CARLOS GÓMEZ LARRAÑAGA (CIMAT)
DR. MIGUEL NAKAMURA SAVOY (CIMAT)
DR. VÍCTOR MANUEL PÉREZ ABREU CARRIÓN (CIMAT)

bajo la presidencia del primero y con carácter de secretario el segundo, para proceder a efectuar el examen que para obtener el grado de

MAESTRO EN CIENCIAS
CON ESPECIALIDAD EN PROBABILIDAD Y ESTADÍSTICA

Sustenta

JESÚS MANUEL PÉREZ ANGULO

en cumplimiento con lo establecido en los reglamentos y lineamientos de estudios de posgrado del Centro de Investigación en Matemáticas, A.C., mediante la presentación de la tesis

" ANÁLISIS TOPOLÓGICO DE DATOS: ROBUSTICIDAD Y ANÁLISIS DE SENSIBILIDAD DE ALGORITMOS "

Los miembros del jurado examinaron alternadamente al (la) sustentante y después de deliberar entre sí resolvieron declararlo (a):

Aprobado

DR. JOSÉ CARLOS GÓMEZ LARRAGAÑA
Presidente

DR. MIGUEL NAKAMURA SAVOY
Secretario

DR. VÍCTOR MANUEL PÉREZ ABREU CARRIÓN
Vocal

CIMAT
DIRECCIÓN GENERAL

DR. JOSÉ ANTONIO STEPHAN DE LA PEÑA MENA
Director General

Agradecimientos

A mis padres Manuel y Angelita, por haberme guiado siempre con rectitud y ayudarme a ser la persona que soy actualmente. Por sus palabras de sabiduría y su consejo siempre preciso. Gracias por nunca dejarme ir a la deriva.

A mis hermanos Sinué y Josué, por estar ahí cada que he necesitado de oídos y manos extras.

A mis tíos Ignacio y Socorro, por las lecciones de vida que me han brindado y ser un pilar importante de la familia.

A Víctor por toda la paciencia que me tuvo, así como la confianza que depositó en mí. Por todo el apoyo que me brindó a lo largo de la maestría y por supuesto por su guía para el desarrollo de esta tesis.

A mis sinodales, los doctores Miguel Nakamura, Rogelio Hasimoto y Fermín Reveles por haberse tomado el tiempo de leer nuestro trabajo y realizar tan importantes observaciones del mismo.

Al Dr. Clément Maria por todo el apoyo brindado durante su estancia en CIMAT, su ayuda fue bastante significativa para aterrizar de manera correcta distintos conceptos que pude haber malinterpretado.

A Jesús Rocha (Chuche) por toda la asistencia técnica y su aguante siempre constante al apoyarme con el uso del equipo de cómputo.

A Cricelio, Irving y Rocío, por las risas compartidas. Crecimos juntos a lo largo de esta etapa y su amistad es algo que permanecerá más allá de CIMAT.

Al Consejo Nacional de Ciencia y Tecnología (CONACyT) y al Centro de Investigación en Matemáticas por su apoyo económico para poder llevar a cabo mis estudios de maestría así como el presente trabajo de tesis.

Índice general

Introducción

Data has a shape and shape matters.

In Big Data, it's not so much about the "big", the real problem is in the complexity of the data.
—**Gunnar Carlsson**

La cantidad de datos que se generan y es posible almacenar hoy en día es enorme, razón por la cual es necesario contar con herramientas matemáticas adecuadas, eficientes y de bajo costo computacional, que sirvan para realizar un análisis correcto de estos datos para extraer información útil de los mismos. Los conjuntos de datos pueden ser complejos debido a su volumen, alta dimensionalidad y "ruido" entre otras características que no son siempre fáciles de percibir. Incluso podemos catalogarlos como complejos en su manejo computacional en cuanto a su eficiencia "real" y el tiempo de ejecución de los algoritmos.

El Análisis Topológico de Datos (ATD) es una herramienta matemática para estudiar la estructura y forma de los datos, la cual surgió hace dos décadas e involucra elementos de topología algebraica, cómputo, y cada vez más de probabilidad y estadística. Ha tenido éxito y relevancia en diversas aplicaciones como detección de tipos de cáncer (DeWoskin et al., 2010; Nicolau et al., 2011; Arsuaga et al., 2012; Seemann et al., 2012), neurociencia (Singh et al., 2008; Dabaghian et al., 2012; Brown y Gedeon, 2012; Romano et al., 2014), tratamiento de imágenes (Carlsson et al., 2008), genética (Bowman et al., 2008; Emmett y Rabadan, 2014), biología (Chan et al., 2013) e incluso en otras ramas de las matemáticas (De Silva y Ghrist, 2007; Horak et al., 2009; Gamble y Heo, 2010), entre otras disciplinas. En particular, en CIMAT se han trabajado aplicaciones de ATD y existen dos trabajos de tesis presentados este año. El primero es el de *Modelos de homología persistente en filogenética* de Ibarra Rodríguez (2016). El segundo es el de *Aspectos Estadísticos en Análisis Topológico de Datos y una Aplicación en Ecología* por González Cucurachi (2016).

El objetivo del ATD es estimar propiedades topológicas de un espacio a partir de una nube de puntos. Es usual suponer que estos datos provienen de una variedad con cierta distribución (usualmente la uniforme) mas un *ruido pequeño*. El objetivo es encontrar "agujeros" $k-$dimensionales en el espacio a través de la homología persistente, la cual es una de las herramientas utilizadas en la topología algebraica con la finalidad de encontrar características topológicas de los espacios de interés. Esta técnica usa como elementos principales a los complejos simpliciales. La idea es analizar la evolución de las características topológicas a medida que cambiamos un parámetro —al cual es usual considerar como *tiempo de filtración*. La evolución respecto a este parámetro se describe como una sucesión

creciente de complejos, lo que se conoce como filtración. La cuantificación de la existencia de los "agujeros" es a través de la estimación de los números de Betti, los cuales son una de las características relevantes del cálculo de la homología persistente. Esta cuantificación se expresa mediante resúmenes topológicos gráficos como los diagramas de persistencia, los códigos de barra y los panoramas de persistencia.

Un primer enfoque de homología persistente para estimar la topología del espacio subyacente a una nube de puntos es mediante los complejos de Čech. El costo computacional en el cálculo de este tipo de complejos puede ser muy alto. El tiempo que toma este análisis depende de distintos factores tales como el tamaño de la muestra, la dimensión del espacio ambiente (dimensión de los datos), la distribución de los datos en la variedad y el valor máximo de la filtración que se le permite tomar al algoritmo. Una primera alternativa para reducir el costo computacional en el cálculo de los complejos de Čech es mediante una relajación en su construcción, la cual se da a través de los complejos Vietoris-Rips (VR). Sin embargo los complejos VR también pueden generar un alto costo computacional que depende de los mismos factores mencionados. De hecho, en ambos algoritmos la complejidad computacional es de orden $O(n^3)$, con n el número de puntos en la nube.

Lo anterior plantea la necesidad de contar con métodos de homología persistente alternativos que no sean tan sensibles al número de puntos en la nube y reduzcan el costo computacional, y que sigan siendo eficientes al explicar la topología de los espacios de interés, particularmente la correspondiente a una nube de puntos. Entre las alternativas existentes se encuentran los complejos alfa introducidos por Edelsbrunner y Mücke (1994). El planteamiento de este método es analizar la estructura de los datos con sólo variar un parámetro de "resolución" α. Una de sus características es que se obtiene el mismo tipo de homotopía que los complejos de Čech pero con un número menor de simplejos. Una segunda alternativa son los complejos testigo propuestos por De Silva y Carlsson (2004), los cuales basan su construcción a partir de una submuestra de la nube de puntos y tomando como "testigos" al resto de la muestra. Esta construcción ayuda a reducir el número de simplejos presentes en el cálculo de la homología así como el costo computacional, siendo éste del orden $O(n^2)$. En Guibas y Oudot (2008) se conjetura que es posible reducir este costo a $O(n \log n)$ bajo algunas condiciones de "dispersión" en los datos.

Otra alternativa son los complejos de Morse-Smale (MS) cuya construcción se basa en los puntos críticos de una función "suave" sobre una variedad. Se usa la teoría de Morse para encontrar tales puntos en variedades de dimensión 0, 1 y 2. Esta teoría fue desarrollada por Morse (1934) primero para dimensiones infinitas, y después para variedades de dimensiones finitas por Milnor (1963). Otra técnica bastante eficiente en cuanto al costo computacional y que describe de buena manera la estructura de los datos es el algoritmo Mapper propuesto por Singh, Mémoli y Carlsson (2007). A diferencia de las otras alternativas mencionadas, Mapper no hace uso de la homología persistente, sin embargo, nos brinda buena intuición acerca de la estructura geométrica de los datos. Mapper se encuentra en una fase avanzada de desarrollo por AYASDI, empresa fundada por Gunnar Carlsson y otros pioneros del ATD. Esta empresa se dedica a realizar análisis de datos complejos mediante este algoritmo, razón por la cual la utilización "completa" no se encuentra abierta a todo público. Una alternativa para conocer el funcionamiento de Mapper son un par de implementaciones en lenguaje Python que se encuentran en la red. Una es

la presentada en Müllner y Babu (2013) cuyo software cuenta con una interfaz gráfica que permite al usuario ajustar los diversos parámetros del algoritmo. También se encuentra un módulo desarrollado por van Veen (2015) el cual utiliza una serie de módulos de Python para obtener la estructura geométrica de los datos. Ambas implementaciones son de libre uso, las instrucciones para su instalación se pueden encontrar en las páginas web de los autores.

Finalmente, una alternativa más fue propuesta recientemente en Hennigan (2015). Esta técnica "combina" ideas que se utilizan en los algoritmos MS y Mapper, construyendo un complejo simplicial a partir de un árbol cuyos nodos son grupos donde la característica que comparten sus elementos es que al proyectarlos sobre un vector principal, el valor del elemento proyectado es menor (o mayor) a la mediana del vector proyectado. Estas componentes se obtienen mediante un método computacional llamado *potencia iterativa* el cual calcula el vector propio respectivo al valor propio mayor en una matriz de distancias. Desafortunadamente el artículo se encuentra en fase de desarrollo y el algoritmo no está completo; sin embargo, decidimos mencionarlo y presentarlo pues la idea promete ser eficiente en tiempo y en aplicación a datos con alta dimensión. Sería interesante lograr su aplicación en un futuro no muy lejano.

Un primer objetivo de este trabajo es presentar un compendio monográfico sobre las distintas metodologías mencionadas anteriormente, presentando numerosos ejemplos de resúmenes topológicos. El trabajo está dirigido a quienes deseen iniciarse en el estudio del ATD y tengan conocimientos previos en estadística. Un material complementario son las notas del curso *Persistencia, Probabilidad e Inferencia Estadística para Análisis Topológico de Datos* impartido por Biscay, Nakamura, Pérez Abreu y Reveles (2016) en el CIMAT en el semestre de primavera de 2016.

Si bien las alternativas descritas tienen un costo computacional menor a los complejos de Čech y VR, es natural la pregunta de qué tan eficientes son estas alternativas para capturar la homología ante diversos escenarios de distribución de la nube de datos, tamaño de la misma, dimensión del espacio ambiente, así como la homología subyacente al espacio de origen de los datos. Un segundo objetivo de esta tesis es evaluar mediante un estudio de simulación la efectividad y sensibilidad de estos algoritmos ante diversas situaciones y comparar su eficacia con el enfoque clásico de los complejos VR. Para este fin se plantearon varios estudios de simulación usando diversas distribuciones "complejas" alternativas a la distribución uniforme usualmente presupuesta, analizando la efectividad y éxito de diversos algoritmos ante escenarios de distintos tamaños de muestra y dimensión del espacio ambiente, en el caso de algunas variedades. En particular, estamos interesados en la robusticidad de los algoritmos ante escenarios de nubes de puntos provenientes de distribuciones diferentes a la distribución uniforme en una variedad.

Con respecto a las implementaciones computacionales que se encuentran disponibles actualmente, la mayoría están en los lenguajes C++ y Python. De manera particular, el paquete TDA de R hace uso de las librerías GUDHI, Dionysus y PHAT desarrolladas en C++. Este paquete nos permite el cálculo de la homología para los complejos VR y MS de cualquier dimensión. En C++ podemos encontrar la librería para calcular la homología de los complejos alfa, la cual es parte de un proyecto de geometría computacional llamado CGAL. Este sólo permite hacer cálculos de la homología para dimensiones (del espacio

ambiente) 2 y 3. Existe una librería de Javaplex que se encuentra implementada como una dependencia de Matlab con la cual es posible calcular los complejos testigos en cualquier dimensión, tanto para datos euclideos como no euclideos. También cuenta con el algoritmo para calcular la persistencia de los complejos VR, pero tiene complicaciones cuando se manejan distribuciones como las que abordamos en este trabajo. En este marco se presenta el estudio de simulación para evaluar la eficiencia de los algoritmos de homología persistente para calcular las propiedades topológicas, así como comparar los tiempos de ejecución y el uso de recursos computacionales en cada caso. Nuestras simulaciones se realizaron en un servidor cuyas características se describen en el Sección 5.2.

La estructura y mayor descripción de los temas de esta tesis son como sigue:

El Capítulo 1 está dedicado a describir el método de simulación de variables aleatorias en algunas variedades con las distintas distribuciones utilizadas en nuestro estudio. Son dos las razones por las que elegimos usar estas distribuciones. En primer lugar, son distribuciones fáciles de simular en las variedades orientables consideradas: esferas y productos cartesianos de éstas. Se incluyen a la esfera $\mathbb{S}^1$ y el toro $\mathbb{T}^2$ que son los modelos de "juguete" usualmente encontrados en estudios de simulación en la literatura del ATD. En segundo lugar, trabajamos con distribuciones más complejas que la uniforme las cuales modelan diferentes aspectos tales como: a) fuertes concentraciones en los ejes cartesianos con y sin "espacios", b) fuerzas de repulsión en los hiperplanos $x_i = x_j$ para $i, j = 1, ..., d$ y c) concentración en diversas regiones de interés. Aspectos estadísticos de esta última distribución han sido considerados recientemente en el artículo de Hernandez-Stumpfhauser, Breidt y van der Woerd (2016). En este capítulo introducimos una aproximación a la distribución uniforme en la esfera $\mathbb{S}^{d-1}$ la cual nos permite permite proponer escenarios de simulación mediante aproximaciones a esta distribución y considerar velocidad de convergencia y la robusticidad (continuidad) de los algoritmos de persistencia que se presentan en los siguientes capítulos. La aproximación se basa en una ley de grandes números que satisfacen varios procesos estocásticos, incluidos los procesos de Lévy. Estos últimos ofrecen diversidad de posibilidades de modelación y son fáciles de simular. El capítulo incluye varios ejemplos de simulación de nubes de datos usando esta aproximación y procesos de Lévy; a este tipo de datos le llamaremos *nubes de datos perturbadas*. Por otro lado, una pregunta natural es qué tan distintas son las nubes de datos simuladas con las distribuciones alternativas respecto a una nube de puntos proveniente de una distribución uniforme, y qué tan distintas son las distribuciones mismas. El Apéndice B presenta estimaciones de distancias apropiadas que se introducen en el Capítulo 2, las cuales miden estas diferencias y son fáciles de calcular.

En el Capítulo 2 presentamos los conceptos de topología algebraica necesarios para estudiar los capítulos siguientes. En la Sección 2.1 damos una breve introducción a grupos y definimos también los grupos cociente. En la Sección 2.2 presentamos los conceptos de espacios topológicos, así como funciones continuas definidas en éstos. En esta misma sección se definen los complejos simpliciales que son la estructura clave del ATD. En la Sección 2.3 se describe la Homología Persistente (HP) que como ya mencionamos, es una herramienta para estimar la topología subyacente de un espacio X dado. Para entender el número de agujeros $k-$dimensionales presentes en los espacios de interés, es posible calcular una aproximación de los números de Betti en un tiempo dado a lo largo de

las filtraciones existentes. Dichas aproximaciones se pueden obtener a través de algunos resúmenes topológicos como los diagramas de persistencia y códigos de barra con los cuales es posible sintetizar la información que nos entrega la HP que presentamos en la Sección 2.4. Por último, en la Sección 2.5 se presentan las nociones teóricas del complejo de Morse-Smale, así como el algoritmo para la construcción de este tipo de complejos. En dicho algoritmo se usa la estimación de densidades vía kernel como un tipo de función "suave". Esta es una primer alternativa a los complejos VR, la cual es posible usar solamente en nubes de datos en espacios euclideos. El capítulo concluye con ejemplos de nubes de datos simuladas con las cuales se compara la efectividad de los algoritmos VR y MS ante diversos escenarios de perturbación de la distribución uniforme generados mediante los métodos de simulación presentados en el Capítulo 1.

En el Capítulo 3 se explican otras alternativas simpliciales a las construcciones de Čech y Vietoris-Rips. En cada caso se presenta el algoritmo, detalles computacionales, así como numerosos ejemplos. En la Sección 3.1 se introducen los complejos Alfa, en los cuales se aprovecha un tipo particular de triangulación del espacio ambiente e intersecciones del mismo con vecindades de los elementos de la nube de puntos. La Sección 3.2 está dedicada a los complejos testigo, cuya idea principal es usar una muestra menor. El capítulo concluye comparando la efectividad de los complejos testigo con los algoritmos VR y MS usando las mismas nubes de puntos perturbadas que se presentan al final del Capítulo 2.

Dedicamos el Capítulo 4 a la presentación del algoritmo Mapper. Este algoritmo usa técnicas de agrupamiento de datos para generar un complejo simplicial que describe la estructura de los mismos. La Sección 4.1 presenta una introducción al agrupamiento jerárquico y algunos de sus algoritmos; éstos se utilizan con la finalidad de reducir el número de simplejos necesarios para describir la estructura de los datos. En la Sección 4.2 se presenta el algoritmo para calcular complejos simpliciales, una de cuyas características es que no hace uso de la homología persistente, concentrándose en la noción geométrica de una nube de datos dada. Finalmente, la Sección 4.3 está dedicada a dos implementaciones en Python de Mapper disponibles en la red, así como algunos ejemplos donde se presenta la interpretación de los resultados que entrega cada una de ellas. El capítulo finaliza con la comparación de la efectividad de Mapper con los algoritmos VR, MS y testigo usando las mismas nubes de datos perturbadas que se presentan al final del Capítulo 2

En el Capítulo 5 se presenta una reseña de un estudio de simulación realizado por (Otter et al., 2015), donde se muestra el desempeño de los algortimos Čech,VR y Alfa. Tomando como base dicho estudio, presentamos los resultados de un estudio de simulación en donde se comparan algunos de los métodos arriba mencionados bajo distintos escenario. Más específicamente, el estudio consiste en analizar el desempeño de estos métodos, variando el tamaño de muestra, así como la cantidad de ruido presente en la nube de datos y la distribución de éstos. Dado que los algoritmos están desarrollados en distintos lenguajes y con diversos ordenes de complejidad computacional, se propuso como criterio de comparación el uso de memoria física, virtual y el tiempo de ejecución de cada uno. De igual manera, es de interés conocer cuál es el tamaño de muestra que soportan los métodos sin colapsar computacionalmente (uso eficiente de los recursos de hardware), así como el valor máximo de la filtración de modo que los algoritmos sigan trabajando y capturen de manera correcta la homología de cada uno de los espacios subyacentes a las nubes de

datos simuladas. En el caso de los complejos testigo elegimos un tamaño de muestra "adecuado" como lo describen De Silva y Carlsson (2004). Todas estas simulaciones se hacen en un "ambiente controlado" pues tenemos conocimiento de las propiedades topológicas y geométricas que nos deben entregar los algoritmos. Los resultados se describen mediante tablas y gráficas comparativas en donde se aprecian los tiempos de cálculo, la cantidad de memoria física y virtual utilizada por cada método. El lector puede consultar en un anexo en línea los resultados obtenidos en el cálculo de la homología (diagramas de persistencia y códigos de barra). Asimismo, al final de este capítulo se presentan gráficas de resultados de simulación que permiten analizar la estabilidad de los algoritmos, respecto al tamaño de muestra y distribuciones cercanas y lejanas de la distribución uniforme.

Finalmente, en el Capítulo 6 se presentan conclusiones generales de todos los trabajos de simulación realizados en esta tesis. Se incluyen ventajas y desventajas de algunos de los algoritmos estudiados en este trabajo, en el marco de los distintos escenarios considerados. Se reafirma el hecho de que estos algoritmos son complementarios. Pensamos que los escenarios propuestos en esta tesis contribuyen a entender dicho complemento. Así mismo, esperamos que estas conclusiones sirvan de referencia a los usuarios que se inician en el uso de los algoritmos de persistencia en el Análisis Topológico de Datos.

En el Apéndice A presentamos los algoritmos para simular datos con perturbaciones a la distribución uniforme en $\mathbb{S}^{d-1}$ mediante procesos de Poisson, gaussiano inverso y normal gaussiano inverso. En el Apéndice B se muestra la comparación de la distancia de Hausdorff entre nubes de datos con las aproximaciones a la distribución uniforme vs. la distribución uniforme, así como la distancia del supremo entre las densidades estimadas asociadas a los mismos. Igualmente, se incluye esta comparación en el caso de la distribución con repulsión vs. la distribución uniforme. Asimismo, se presentan simulaciones de nubes de datos con distribución con repulsión en las variedades $\mathbb{S}^1$, $\mathbb{S}^2$ y $\mathbb{T}^2$ y los diagramas de persistencia respectivos a los algoritmos VR, MS y Alfa. El objetivo de estas simulaciones es ilustrar la efectividad de estos algoritmos para detectar la topología subyacente, así como comparar de manera visual diagramas de persistencia asociados a esta distribución. Finalmente el Apéndice C se incluyen numerosos ejemplos donde se muestran perturbaciones a la distribución uniforme en $\mathbb{S}^1$, $\mathbb{S}^2$ y $\mathbb{T}^2$ y sus posibilidades de modelación de nubes de datos con ciertas características.

1

Ciertas distribuciones en la esfera y el toro: Simulación

La mayoría de los estudios de simulación que se encuentran en la literatura de ATD para ilustrar y evaluar algoritmos y métodos han sido sobre variedades como la esfera y el toro 2-dimensional (en adelante simplemente diremos *toro*), considerando solamente la distribución uniforme sobre éstas. En este capítulo se presentan distribuciones sobre dichas variedades que son usadas en los estudios de simulación en esta tesis, las cuales son más complejas que la distribución uniforme. En la Sección 1.1 se proponen algunas distribuciones que modelan diversos aspectos de interés, como fuerte concentración en ejes cartesianos, fuerzas de repulsión en algunos hiperplanos y concentración en regiones de interés. Por otro lado en la Sección 1.2 presentamos un resultado sobre aproximación o perturbación de distintas distribuciones en esferas, el cual usa procesos estocásticos. En la Sección 1.2.1 ilustramos la propuesta con varios procesos de Lévy que dan flexibilidad de modelación y son fáciles de simular. El objetivo es analizar mediante simulación la robusticidad y sensibilidad de los algoritmos de persistencia que se consideran en los siguientes capítulos, en un escenario de control de la aproximación o perturbación. En la Sección 1.2.2 se muestran ejemplos de diversos escenarios de procesos donde se observa el comportamiento de las perturbaciones y su convergencia. Usando las distancias entre nubes de puntos y entre distribuciones que se presentan en el Capítulo 2, el Apéndice B contiene un estudio empírico para comparar distancias entre nubes de datos provenientes de una distribución uniforme y nubes de datos simuladas con las distribuciones alternativas que se proponen en este capítulo, así como distancias entre las distribuciones mismas. La elección de estas distancias se debe a que se encuentran implementadas en R.

1.1. Distribuciones cociente en $\mathbb{S}^{d-1}$

Una *variedad* es un objeto que visto "de cerca" tiene un comportamiento geométrico similar a $\mathbb{R}^d$. Las $d-1$ esferas $\mathbb{S}^{d-1}$ en $\mathbb{R}^d$ son ejemplos de variedades. Existe una manera fácil de simular variables aleatorias uniformes en el círculo $\mathbb{S}^1 = \left\{(x,y) \in \mathbb{R}^2 : x^2 + y^2 = 1\right\}$, el cual consiste en parametrizar $\mathbb{S}^1 = \left\{e^{i\theta} : 0 \leqslant \theta \leqslant 2\pi\right\}$ y generar θ mediante una distribución uniforme en $[0, 2\pi]$. Sin embargo, se puede presentar el caso de que a pesar de simular de manera uniforme sobre los parámetros de una variedad no se preserve la distribución uniforme en la variedad, ver por ejemplo las notas de Biscay et al. (2016, Capítulos 3 y 4).

Una manera alternativa de generar variables aleatorias sobre $\mathbb{S}^{d-1} \subset \mathbb{R}^d$ es considerando un vector $\mathbf{X} = (X_1, ..., X_d)$ con distribución μ en $(\mathbb{R}^d, \mathcal{B}(\mathbb{R}^d))$ tal que $\mathbb{P}(\|\mathbf{X}_d\| = 0) \neq 0$. Entonces

$$S_d = \left(\frac{X_1}{\|\mathbf{X}\|}, ..., \frac{X_d}{\|\mathbf{X}\|}\right), \tag{1.1}$$

es una variable aleatoria en $\mathbb{S}^{d-1}$ cuya distribución es la medida inducida

$$\mu_{S_d}(A) = \mu\left(S_d^{-1}(A)\right), \qquad A \in \mathcal{B}(\mathbb{S}^{d-1}).$$

Nos referimos a esta construcción como distribución cociente $\mathbb{R}^d/\mathbb{S}^{d-1}$ o simplemente *distribución cociente* en $\mathbb{S}^{d-1}$. Para mayores detalles de este tema referimos al lector a Biscay et al. (2016).

Este método nos permite simular variables aleatorias en $\mathbb{S}^{d-1}$ con distintas distribuciones alternativas a la uniforme. Asimismo, es posible añadir ruido a estas variables aleatorias al sumarle una normal d variada, siguiendo el modelo

$$S_d + \sigma N_d(0, I_d),$$

donde el parámetro σ nos permite controlar la cantidad de ruido en la muestra. En las ilustraciones de las distribuciones en $\mathbb{S}^{d-1}$ que hacemos en este trabajo de tesis utilizamos un valor pequeño de σ, es en esta situación que nos referimos a *ruido pequeño*. A continuación presentamos ejemplos de distribuciones que son usadas en los estudios de simulación en esta tesis.

1.1.1. Distribución cociente uniforme

Si $\mathbf{X} = (X_1, ..., X_d)$ es un vector aleatorio con distribución normal multivariada $N_d(0, I_d)$, entonces se satisface que S_d tiene la distribución uniforme sobre $\mathbb{S}^{d-1}$. La siguiente figura muestra simulaciones de 1,000 variables aleatorias con distribución uniforme en $\mathbb{S}^1$ y $\mathbb{S}^2$.

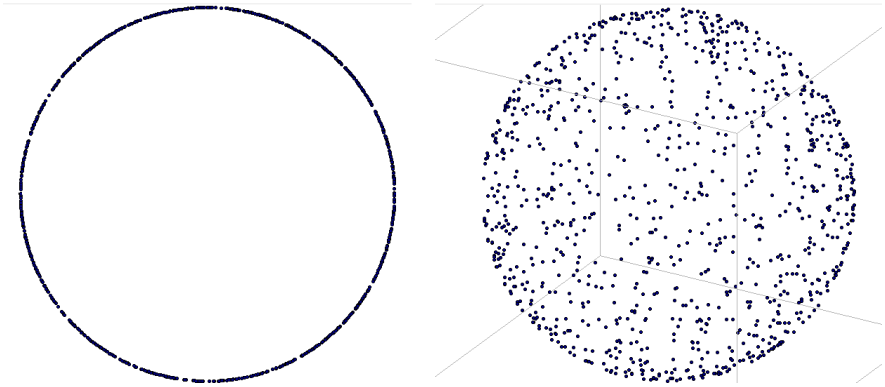


Figura 1.1: Simulación de 1000 variables aleatorias con distribución uniforme en $\mathbb{S}^1$ y $\mathbb{S}^2$.

1.1.2. Distribución cociente concentrada en regiones

Sea $\mathbf{X} = (X_1, ..., X_d)$ un vector aleatorio con distribución normal multivariada $N_d(m, \Sigma)$. Si $m \neq 0$ y $\Sigma = (\sigma_{ij})$ no es un múltiplo de la matriz identidad I_d, la distribución de S_d tiende a concentrarse en algunas regiones sobre la intersección de los hiperplanos "identidad" con la esfera $\mathbb{S}^{d-1}$. La concentración se da en mayor o menor escala dependiendo de los valores de las correlaciones σ_{ij}. En las Figuras 1.2 y 1.3 mostramos distintos comportamientos correspondientes a diversas correlaciones para los casos de $\mathbb{S}^1$ y $\mathbb{S}^2$.

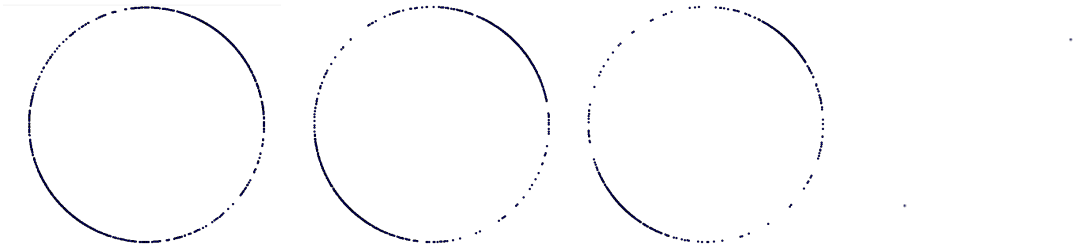


Figura 1.2: Simulación de 1000 variables aleatorias en $\mathbb{S}^1$ con correlaciones $0.8, 0.95, 0.99, 1$ respectivamente.

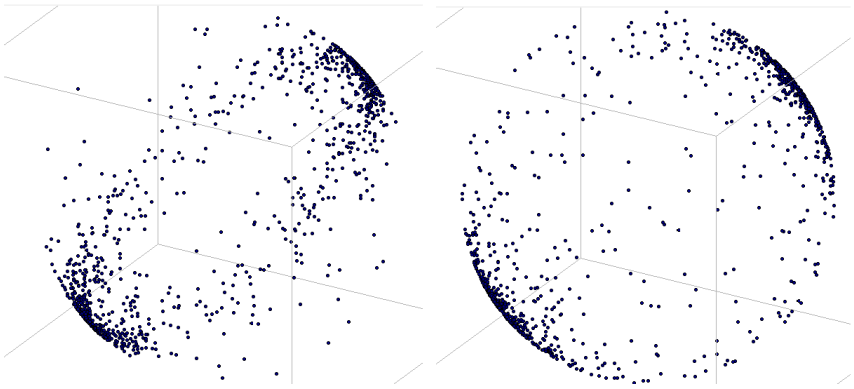


Figura 1.3: Simulación de 1,000 variables aleatorias sobre $\mathbb{S}^2$ con $\sigma_{12} = 0.9, \sigma_{13} = 0.5, \sigma_{23} = 0.8$ y $\sigma_{12} = 0.9, \sigma_{13} = 0.9, \sigma_{23} = 0.9$, respectivamente

1.1.3. Distribución cociente concentrada en ejes cartesianos

Si $\mathbf{X} = (X_1, ..., X_d)$ satisface que cada una de sus entradas X_i, $i = 1, ..., d$, son independientes y tienen distribución Cauchy$(0, 1)$, se tiene que la distribución S_d se concentra en los puntos de intersección de la esfera con los ejes cartesianos. En este caso, la distribución Cauchy tiene un comportamiento similar al de la distribución normal, excepto que al tratarse de una distribución de colas pesadas el comportamiento de la distribución tiende a dar las concentraciones en los puntos descritos. En las Figuras 1.4 y 1.5 damos ejemplos, la primera sin ruido y en la segunda con un ruido pequeño que nos ayuda a distinguir de manera visual la concentración en los puntos cartesianos de $\mathbb{S}^1$ y $\mathbb{S}^2$.

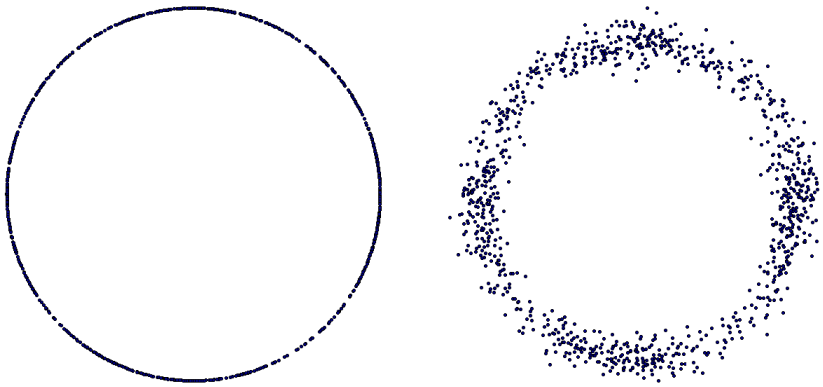


Figura 1.4: Simulación de 1,000 variables aleatorias con distribución concentrada en la intersección de los ejes cardinales con la variedad $\mathbb{S}^1$.

En el caso de $\mathbb{S}^2$ tenemos un comportamiento similar:

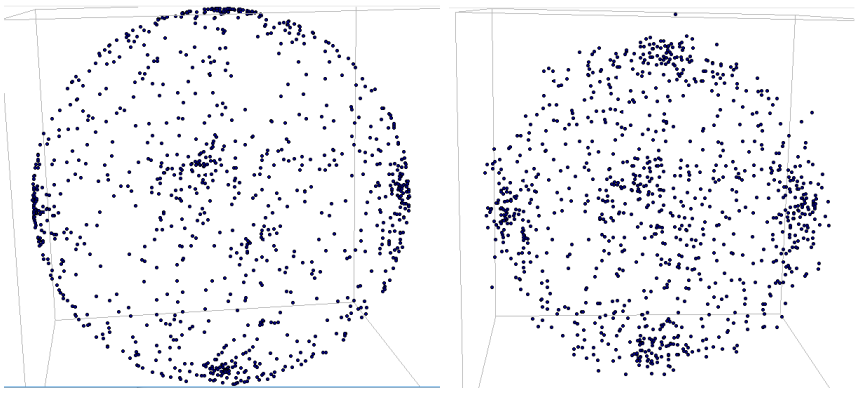


Figura 1.5: Simulación de 1,000 variables aleatorias con distribución concentrada en la intersección de los ejes cartesianos con la variedad $\mathbb{S}^2$.

1.1.4. Distribución cociente con repulsión en hiperplanos

Consideremos la matriz aleatoria hermitiana $\mathbf{Z} = (Z_{ij})$ de tamaño $d \times d$, donde $\mathrm{Re}(Z_{ij})$ e $\mathrm{Im}(Z_{ij})$ para $1 \leqslant i, j \leqslant d$ son variables aleatorias independientes con distribución $N(0, \frac{1}{2}(1 + \delta_{ij}))$. A esta matriz Z se le llama matriz GUE (Gaussian Unitary Ensemble).

La densidad multivariada de los vectores propios $X_1, ..., X_d$ de la matriz Z está dada por

$$f(\mathbf{x}) = \tilde{c}_d \exp\left(-\frac{1}{2}\|\mathbf{x}\|\right) \prod_{i<j} |x_i - x_j|^2, \qquad \text{con } \mathbf{x} = (x_1, ..., x_d) \in \mathbb{R}^d,$$

donde $\tilde{c}_d$ es una constante positiva que sólo depende de la dimensión d.

En este caso podemos observar que no podemos descomponer la densidad como producto de densidades marginales de cada entrada x_i y más aún, éstos son fuertemente dependientes y tienen una fuerza de repulsión. Es por tanto que esta fuerza de repulsión se ve reflejada en la distribución de S_d. En la Figura 1.6 ilustramos la repulsión que se genera en S_d bajo esta distribución.

Figura 1.6: Simulación de 1,000 variables aleatorias con distribución cociente con repulsión GUE en $\mathbb{S}^1$ y $\mathbb{S}^2$.

1.1.5. Distribuciones en productos cartesianos $\mathbb{S}^n \times \mathbb{S}^m$

En la Sección 1.1 vimos cómo simular variables aleatorias con distribución cociente en las $d-1$ esferas $\mathbb{S}^{d-1}$. Usando la medida producto es posible simular variables aleatorias en productos cartesianos de esferas, un caso particular es en el toro $\mathbb{T}^2 = \mathbb{S}^1 \times \mathbb{S}^1$. De manera más amplia, es posible generar variables aleatorias en un $k-$toro haciendo el producto cartesiano $\mathbb{T}^k = \underbrace{\mathbb{S}^1 \times \cdots \mathbb{S}^1}_{k \text{ veces}}$.

Un ejemplo se muestra en la Figura 1.7, donde se simulan variables aleatorias con distribución uniforme en $\mathbb{T}^2 = \mathbb{S}^1 \times \mathbb{S}^1$. Si bien esta variedad simulada está en dimensión 4, se utiliza una proyección sobre $\mathbb{R}^3$.

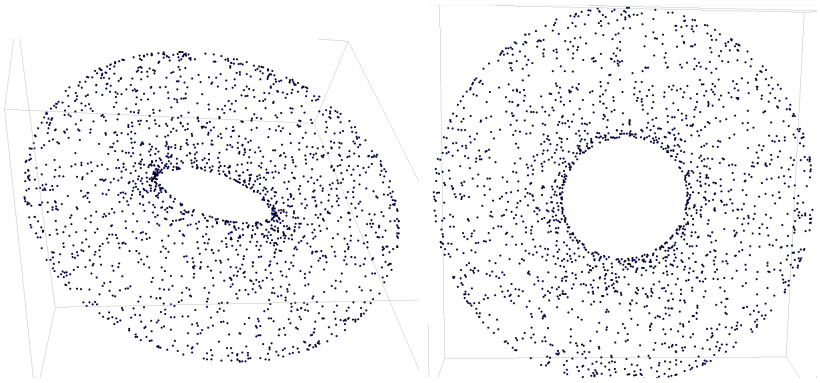


Figura 1.7: Simulación de 2000 variables aleatorias con distribución cociente uniforme sobre $\mathbb{T}^2$.

Más específicamente, estamos utilizando la función $f : \mathbb{T}^2 \subset \mathbb{R}^4 \longrightarrow \mathbb{R}^3$ definida como

$$f(x, y, w, z) = (0, 0, w) + (z + 2)(x, y, 0).$$

Dado que en $\mathbb{S}^{d-1}$ podemos simular variables aleatorias con distintas distribuciones, es de esperarse que en los productos cartesianos se obtenga un comportamiento similar. Ejemplificamos esto en la Figura 1.8, donde se utiliza la distribución concentrada en ejes cartesianos sobre $\mathbb{S}^1$.

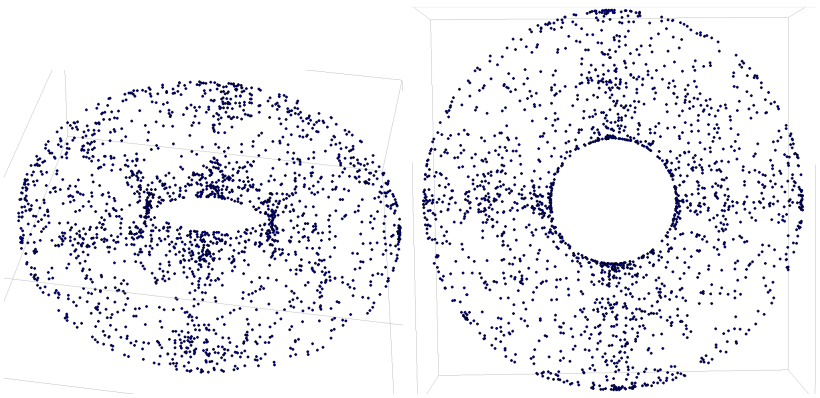


Figura 1.8: Simulación de 2000 variables aleatorias con distribución cociente concentrada en los ejes cartesianos sobre $\mathbb{T}^2$.

1.2. Aproximaciones a la distribución uniforme en $\mathbb{S}^{d-1}$

En esta sección presentamos un resultado general de convergencia de variables aleatorias en esferas a una distribución dada, la cual es perturbada mediante cambios de escala inducidos por un proceso estocástico. El objetivo es contar con posibilidades de modelación alternativas a la distribución uniforme.

Teorema 1. *Sean $\sigma_i(t)$, $i = 1, ..., d$ procesos estocásticos reales con segundo momento finito, tales que existe una función no aleatoria $R_t > 0$ con*

$$\frac{\sigma_i(t)}{R_t} \xrightarrow{Pr} a_i \text{ cuando } t \to \infty, \forall i \quad y \quad a_i \in \mathbb{R}\backslash\{0\}. \tag{1.2}$$

Sean $Y_1, Y_2, ..., Y_d$ variables aleatorias, entonces:

$$S_d^t = \frac{(\sigma_1(t)Y_1, \sigma_2(t)Y_2, \ldots, \sigma_d(t)Y_d)}{\sqrt{(\sigma_1(t)Y_1)^2 + \cdots + (\sigma_d(t)Y_d)^2}} \xrightarrow[t\to\infty]{\mathcal{L}} \frac{(a_1Y_1, \ldots, a_dY_d)}{\sqrt{a_1^2Y_1^2 + \cdots a_d^2Y_d^2}} = S_d^\infty = S_d. \tag{1.3}$$

Para la demostración de este teorema, usamos los siguientes dos resultados clásicos de convergencia de variables aleatorias, los cuales incluimos por completez.

En lo que sigue $\xrightarrow{\text{Pr}}$ y $\xrightarrow{\mathcal{L}}$ denotan convergencia en probabilidad y distribución, respectivamente.

Resultado 1. Sean X_n, X, Y_n, Y $(n \geqslant 1)$ variables aleatorias. Sea g una función continua.

(i) Si $X_n \xrightarrow{\text{Pr}} X$ y $Y_n \xrightarrow{\text{Pr}} Y$, entonces $X_n + Y_n \xrightarrow{\text{Pr}} X + Y$.

(ii) Si $X_n \xrightarrow{\text{Pr}} X$ y $Y_n \xrightarrow{\text{Pr}} Y$, entonces $X_n \cdot Y_n \xrightarrow{\text{Pr}} X \cdot Y$.

(iii) Si $X_n \xrightarrow{\text{Pr}} X$, entonces $g(X_n) \xrightarrow{\text{Pr}} g(X)$.

(iv) Si $X_n \xrightarrow{\text{Pr}} X$, entonces $X_nY \xrightarrow{\text{Pr}} XY$.

Proposición 2 (Cramér-Wold Device)**.** *Si $\{Y_n\}_{n=1}^\infty$ es una sucesión de vectores aleatorios $d-$dimensionales que satisfacen que $c'Y_n \xrightarrow{\mathcal{L}} c'Y$ cuando $n \to \infty$, para todo $c \in \mathbb{R}^d$. Entonces $Y_n \xrightarrow{\mathcal{L}} Y$.*

Prueba del Teorema 1. Como $\frac{\sigma_i(t)}{R_t} \xrightarrow{\text{Pr}} a_i$, y al ser Z_i una variable aleatoria se tiene que

$$\frac{\sigma_i(t)}{R_t} Z_i \xrightarrow{\text{Pr}} a_i Z_i \qquad \text{para cada } i.$$

Sea $X_t = \left(\frac{\sigma_1(t)}{R_t}Y_1, ..., \frac{\sigma_d(t)}{R_t}Y_d\right)$, entonces tenemos que para cualquier vector de constantes arbitrario $c = (c_1, ..., c_d) \in \mathbb{R}^d$

$$c^tY = \frac{\sigma_1(t)}{R_t}c_1Y_1 + \cdots + \frac{\sigma_d(t)}{R_t}c_dY_d \xrightarrow{\text{Pr}} c_1a_1Y_1 + \cdots + c_da_dY_d = c^TY, \text{ cuando } t \to \infty.$$

De manea análoga, como $\left(\frac{\sigma_i(t)}{R_t}\right)^2 \xrightarrow{\Pr} a_i^2$, se puede ver que:

$$g(\mathbf{1}^T Y_t) = \sqrt{(\sigma_1(t)Y_1)^2 + \cdots + (\sigma_d(t)Y_d)^2} \xrightarrow{\Pr} \sqrt{a_1^2 Y_1^2 + \cdots a_d^2 Y_d^2} = g(\mathbf{1}^T Y)$$

cuando $t \to \infty$.

De este modo,

$$c^T Y_t g(\mathbf{1}^T Y_t) \xrightarrow{\Pr} c^T Y g(\mathbf{1}^T Y),$$

lo cual implica el resultado:

$$\frac{(\sigma_1(t)Y_1, \sigma_2(t)Y_2, \ldots, \sigma_d(t)Y_d)}{\sqrt{(\sigma_1(t)Y_1)^2 + \cdots + (\sigma_d(t)Y_d)^2}} \xrightarrow{\mathcal{L}} \frac{(a_1 Y_1, \ldots, a_d Y_d)}{\sqrt{a_1^2 Y_1^2 + \cdots a_d^2 Y_d^2}}.$$

Simplemente resta dividir numerador y denominador por R_t. □

La siguiente proposición además de ser una manera de realizar aproximaciones a las cuatro distribuciones presentadas en la sección anterior, puede también pensarse como una forma de perturbación en dichas distribuciones, siendo el parámetro de perturbación el tiempo t. De hecho, el caso (a) con $Y_1, ..., Y_d$ variables independientemente distribuidas $N(0,1)$ y $a_i = a \neq 0$ en (1.2) para todo i, puede ser pensado como un Teorema del Límite Central, donde la distribución límite es la uniforme en $\mathbb{S}^{d-1}$.

Proposición 3. *Sean $\sigma_1(t), \sigma_2(t), ..., \sigma_d(t)$ procesos estocásticos tales que existe un proceso no estocástico R_t que satisface que $\sigma_i(t)/R_t \xrightarrow{Pr} a, a \in \mathbb{R}\setminus\{0\}$ para todo $i = 1, ..., d$.*

(a) Si $Y_1, ..., Y_d$ son variables aleatorias independientes con distribución $N(0,1)$, S_d tiene la distribución uniforme en $\mathbb{S}^{d-1}$.

(b) Si $Y_1, ..., Y_d$ son los valores propios de una matriz aleatoria GUE, S_d tiene la distribución con repulsión en hiperplanos en $\mathbb{S}^{d-1}$.

(c) Si $Y_1, ..., Y_d$ son variables aleatorias independientes con distribución Cauchy$(0,1)$, entonces S_d tiene la distribución concentrada en los ejes cartesianos en $\mathbb{S}^{d-1}$.

Una clase rica de procesos estocásticos, los cuales son sencillos de simular y en los que además se cumple fácilmente la Ley de Grandes Números (1.2) son los procesos de Lévy, los cuales se consideran a continuación. Nuestro objetivo es utilizar estos procesos y la última proposición para obtener aproximaciones a las distribuciones en $\mathbb{S}^{d-1}$ presentadas en la sección anterior. En particular presentamos varios ejemplos de convergencia a la distribución uniforme en $\mathbb{S}^{d-1}$ y $\mathbb{T}^2$. Posteriormente se presentan varios ejemplos en la Sección 2.6.

1.2.1. Procesos de Lévy

Los procesos de Lévy son procesos estocásticos bien estudiados, que permiten diversas y flexibles posibilidades de modelación (ver Kyprianou (2006)). En particular, para nuestros estudios de simulación, estamos interesados en considerar procesos con diversas posibilidades de modelación como los que toman valores y saltos discretos, no negativos

o reales, al mismo tiempo que distribuciones con colas pesadas, que sean alternativas a la Cauchy considerada anteriormente, pero que tengan media y varianza finitas. Así mismo, algunos de estos procesos son fáciles de simular.

Un proceso estocástico real valuado, continuo por la derecha y con límites izquierdos $L = \{L_t, t \geqslant 0\}$ con $L_0 = 0$ c.s. se llama *proceso de Lévy* si se satisfacen las siguientes condiciones:

- L tiene *incrementos independientes*, es decir $L_t - L_s$ es independiente de $\mathcal{F}_s$ para cualesquiera $0 \leqslant s < t \leqslant T$, donde $\mathcal{F}_s$ es la $\sigma-$álgebra generada hasta el tiempo s. Es decir, el incremento $L_t - L_s$ es independiente de lo que ocurre antes del tiempo s.

- L tiene *incrementos estacionarios*, es decir, para cualesquiera $s, t \geqslant 0$ la distribución de $L_{t+s} - L_t$ no depende de t.

- L es *estocásticamente continuo*, es decir, para todo $t > 0$ y $\varepsilon > 0$ se tiene que

$$\lim_{s \to t} \mathbb{P}\left(|L_t - L_s| > \varepsilon\right) = 0.$$

La distribución de un proceso de Lévy $\{L_t\}_{t \geq 0}$ es tal que para cada $t > 0$ la función característica $\phi_t(u)$ de L_t tiene la forma $\phi_t(u) = \exp(t\Psi(u))$, donde el exponente característico $\Psi(u)$ satisface la representación de Lévy-Khintchine:

$$\Psi(u) = i\gamma u - \frac{1}{2}\sigma^2 u^2 + \int_{-\infty}^{\infty} \left(e^{iux} - 1 - iux\mathbb{1}_{\{|x|\}<1}\right) \nu(dx), \tag{1.4}$$

donde $\gamma \in \mathbb{R}$, $\sigma^2 \geqslant 0$ y ν es una medida en $\mathbb{R}\setminus\{0\}$ tal que

$$\int_{\mathbb{R}\setminus\{0\}} \text{mín}\left\{x^2, 1\right\} \nu(dx) < \infty.$$

De este modo, un proceso de Lévy está caracterizado por la terna (γ, σ^2, ν), cuya interpretación es la siguiente. El parámetro γ controla la deriva del proceso, σ es la parte gaussiana y la *medida de Lévy* ν modela los saltos del proceso.

A continuación presentamos ejemplos de cuatro procesos de Lévy que usaremos en esta tesis.

1.2.1.1. Proceso Poisson

Un proceso de Lévy $N(t)$ se llama *proceso de Poisson con tasa* $\lambda > 0$ si para cada $t > 0$, la distribución de $N(t)$ es Poisson con parámetro λt cuya función de probabilidad esta dada por:

$$\mathbb{P}(N(t) = n) = \frac{(\lambda t)^n}{n!} e^{-\lambda t}, \qquad n = 0, 1, 2, ...$$

La esperanza y la varianza son $\mathbb{E}(N(t)) = \lambda t$ y $\mathrm{Var}(N(t)) = \lambda t$. Recordemos que este proceso toma valores en los enteros no negativos. Una trayectoria de este proceso se observa en la Figura 1.9.

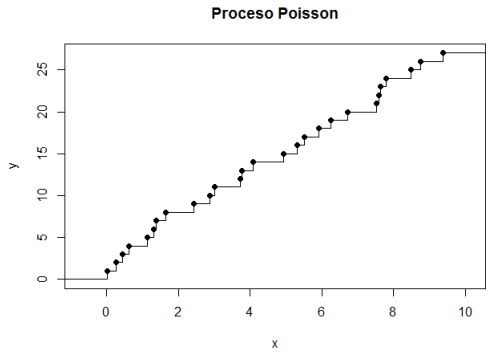


Figura 1.9: Proceso Poisson con parámetro $\lambda = 4$, para $t = 10$.

1.2.1.2. Movimiento browniano

Un movimiento browniano unidimensional de parámetro σ^2 es un proceso de Lévy $\{B_t : t \geqslant 0\}$ con trayectorias continuas y tal que para cualquier tiempo $t > 0$ se tiene que B_t tiene distribución $N(0, \sigma^2 t)$, es decir, su densidad está dada por:

$$f_t(x) = \frac{1}{\sqrt{2\pi t\sigma^2}} e^{-\frac{x^2}{2t\sigma^2}}, \qquad x \in \mathbb{R}.$$

La esperanza del proceso es $\mathbb{E}(B_t) = 0$ y la varianza es $\mathrm{Var}(B_t) = t\sigma^2$. Entre todos los procesos de Lévy, es el único proceso continuo, todos los demás tienen saltos. En la Figura 1.10 se muestra un ejemplo de un movimiento browniano.

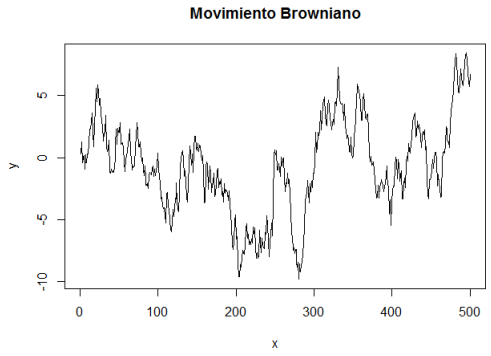


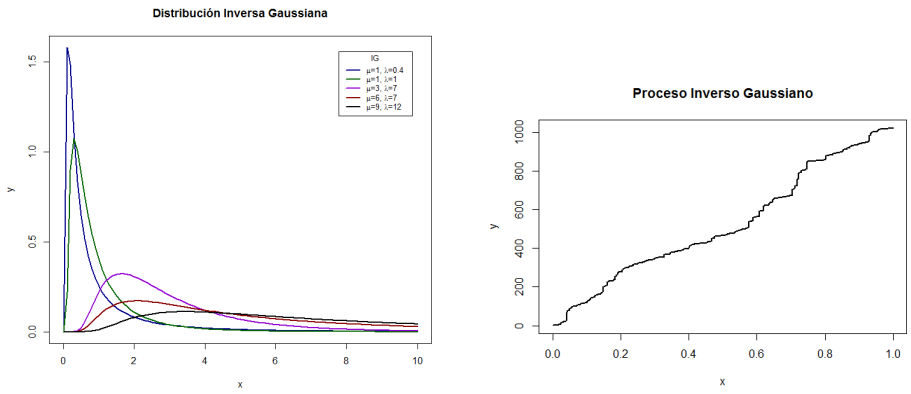
Figura 1.10: Movimiento Browniano.

1.2.1.3. Proceso gaussiano inverso

Un proceso Gaussiano Inverso (IG), I de parámetros λ y μ es un proceso de Lévy tal que para cada $t > 0$, $I(t)$ tiene una distribución inversa gaussiana $\mathcal{IG}(\lambda t^2, \mu t)$ con densidad:

$$f_t(x) = \sqrt{\frac{\lambda t^2}{2\pi x^3}} \exp\left[-\frac{\lambda t^2 (x-\mu t)^2}{2\mu^2 t^2 x}\right], \qquad x > 0.$$

La esperanza del proceso es $\mathbb{E}(I(t)) = \mu t$ y la varianza es $\text{Var}(I(t)) = \mu^3 t/\lambda$. La distribución inversa gaussiana es una distribución continua de dos parámetros la cual tiene la interpretación de ser la distribución del primer tiempo de paso a un nivel $\lambda > 0$ de un movimiento Browniano. El parámetro μ es la media y λ es el parámetro de forma de la distribución. Se trata entonces de un proceso de Lévy con valores en los números no negativos y los saltos del proceso son también no negativos. Sus parámetros permiten modelar las colas de la distribución como se muestra en la Figura 1.11. Es decir, es posible modelar colas pesadas y no pesadas.



(a) Gráficas de densidades IG para distintos parámetros con $t = 1$.

(b) Proceso gaussiano inverso de parámetros $\mu = 1$, $\lambda = 20$.

Figura 1.11: Simulación de densidades y un proceso Gaussiano Inverso.

1.2.1.4. Proceso normal gaussiano inverso

Un proceso Normal Gaussiano Inverso (NIG) $\mathcal{N}$ de parámetros $\mu, \alpha, \beta, \delta$, es un proceso de Lévy tal que la distribución del proceso $\mathcal{N}(t)$, para cualquier tiempo $t > 0$, sigue una distribución normal inversa gaussiana cuya densidad está dada por

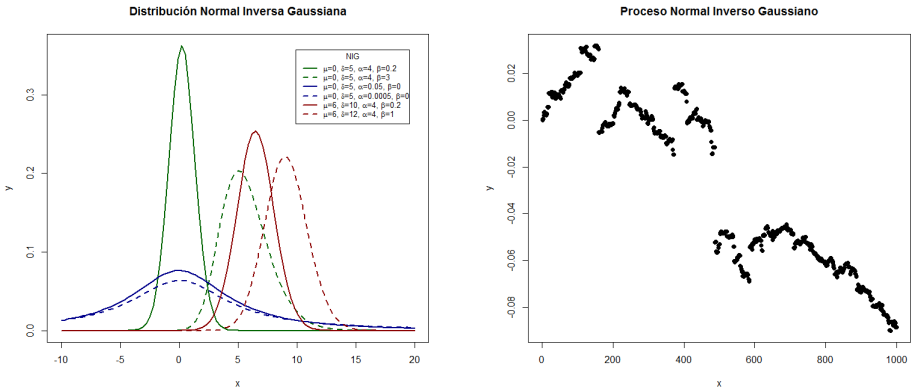
$$f_t(x) = \frac{\alpha\delta t K_1\left(\alpha\sqrt{\delta^2 + (x-\mu t)^2}\right)}{\pi\sqrt{\delta^2 + (x-\mu t)^2}} e^{\delta t\sqrt{\alpha^2-\beta^2}+\beta(x-\mu t)}, \; x \in \mathbb{R},$$

donde μ es el parámetro de localización (por conveniencia se toma $\mu = 0$), α indica lo pesado de las colas de la distribución, β indica la simetría y δ es el parámetro de escala. Además, $K_j(z)$ es la función de Bessel modificada de primer orden del tercer tipo (Campbell, 1980),

$$K_j(z) = \frac{1}{2}\int_0^\infty u^{j-1}\mathrm{e}^{-\frac{1}{2}z(u+u^{-1})}du, \qquad z > 0.$$

La esperanza del proceso NIG es $\mathbb{E}(\mathcal{N}) = t\,(\mu + \beta/\gamma)$ y la varianza del mismo es $\mathrm{Var}(\mathcal{N}) = \alpha^2\delta t/\gamma^3$, donde $\gamma = \sqrt{\alpha^2 - \beta^2}$.

Este proceso tiene saltos tanto negativos como positivos y de acuerdo a los parámetros elegidos puede modelar una distribución con colas pesadas. En la siguiente figura se muestran algunos ejemplos del comportamiento de las densidades para diferentes parámetros así como trayectorias del proceso NIG.



(a) Gráficas de la densidad NIG para distintos parámetros con $t = 1$.

(b) Proceso gaussiano inverso de parámetros $\alpha = 9$, $\beta = 5$, $\delta = 0.002$.

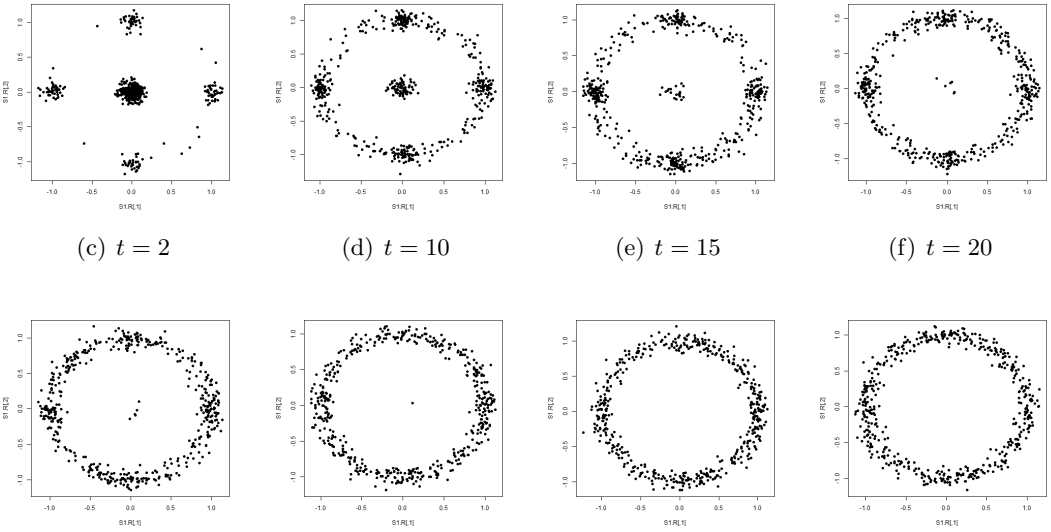
Usando la desigualdad de Chebyshev, se puede probar fácilmente que en el caso de los ejemplos de procesos de Lévy anteriores (excepto en el Browniano) se tiene la convergencia de las aproximaciones S_d^t a la distribución uniforme en $\mathbb{S}^{d-1}$ tomando $R_t = \mathbb{E}(\sigma_1(t)) = \mathbb{E}(\sigma_2(t)) = \cdots = \mathbb{E}(\sigma_d(t))$ en la Proposición 3.

En el Apéndice A presentamos los algoritmos para simular variables aleatorias en $\mathbb{S}^d$ para los ejemplos de procesos de Lévy que se presentan en esta sección.

1.2.2. Ejemplos

A continuación se presentan ejemplos para ilustrar la convergencia de las aproximaciones obtenidas mediante la Proposición 3(a) a la distribución uniforme en $\mathbb{S}^1$, $\mathbb{S}^2$ y $\mathbb{T}^2$. El primero es una aproximación a la distribución uniforme en $\mathbb{S}^1$ perturbando ambas entradas con procesos de Poisson de parámetro $\lambda = 0.1$, mientras que el segundo es una perturbación en la primera entrada usando un proceso inverso gaussiano de parámetros $\mu = 0.1$ y $\lambda = 0.01$. El tercer ejemplo es una aproximación en $\mathbb{S}^2$ donde la perturbación de las tres entradas es mediante procesos inversos gaussianos de parámetros $\mu = 0.1$ y $\lambda = 0.0001$. El último ejemplo es una aproximación en un toro $\mathbb{T}^2$, donde cada $\mathbb{S}^1$ del producto cartesiano que lo genera está perturbado en la primer entrada por un proceso Poisson y la segunda entrada por un proceso gaussiano inverso, los parámetros son ta-

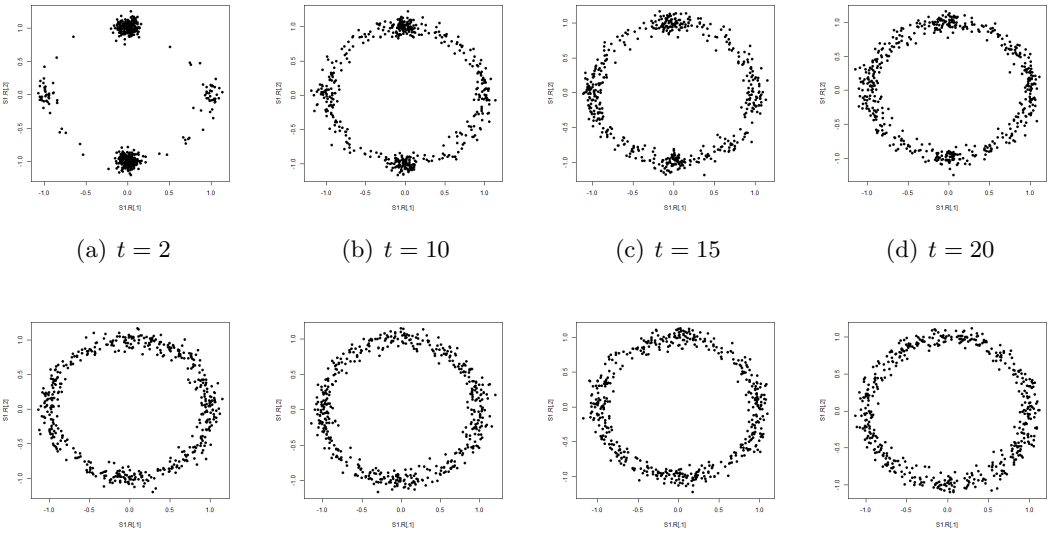
les que la media y la varianza sean iguales en ambos casos $\mathbb{E}(N_1(t)) = \mathbb{E}(I_2(t)) = 0.1$, $\mathrm{Var}(N_1(t)) = \mathrm{Var}(I_2(t)) = 10$.



(c) $t = 2$ (d) $t = 10$ (e) $t = 15$ (f) $t = 20$

(g) $t = 25$ (h) $t = 30$ (i) $t = 40$ (j) $t = 50$

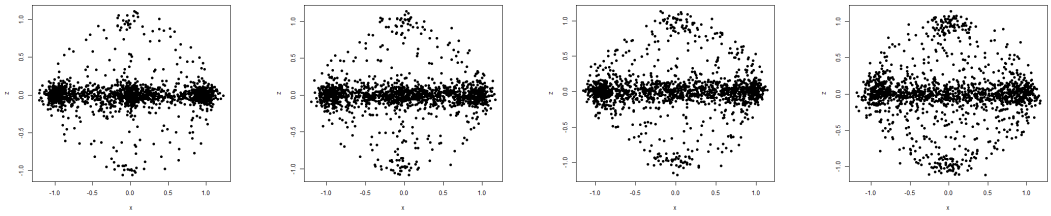
Figura 1.12: 500 datos sobre $\mathbb{S}^1$ con coeficientes de perturbación mediante procesos de Poisson $N_1(t)$ y $N_2(t)$ de parámetros $\lambda_1 = \lambda_2 = 0.1$.



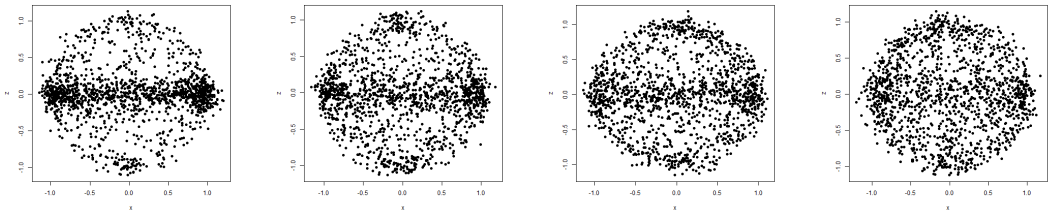
(a) $t = 2$ (b) $t = 10$ (c) $t = 15$ (d) $t = 20$

(e) $t = 25$ (f) $t = 30$ (g) $t = 40$ (h) $t = 50$

Figura 1.13: 500 datos sobre $\mathbb{S}^1$ con coeficientes de perturbación mediante procesos de Poisson $N_1(t)$ y gaussiano inverso $I_2(t)$ con $\mathbb{E}(N_1(t)) = \mathbb{E}(I_2(t)) = 0.1$ y $\mathrm{Var}(N_1(t)) = \mathrm{Var}(I_2(t)) = 0.1$.

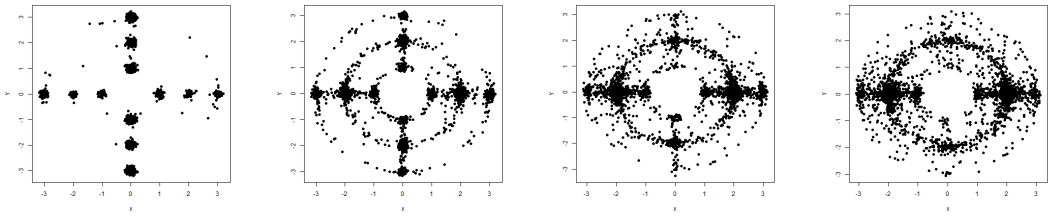


(a) $t = 2$ (b) $t = 10$ (c) $t = 25$ (d) $t = 50$

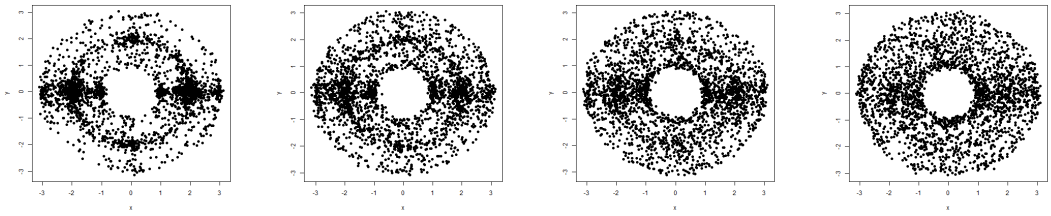


(e) $t = 100$ (f) $t = 300$ (g) $t = 500$ (h) $t = 1000$

Figura 1.14: 1500 datos sobre $\mathbb{S}^2$ con coeficientes de perturbación mediante procesos gaussiano inverso $I_1(t)$,$I_2(t)$ y $I_3(t)$ con $\mathbb{E}(I_1(t)) = \mathbb{E}(I_2(t)) = \mathbb{E}(I_3(t)) = 0.1$ y $\text{Var}(I_1(t)) = \text{Var}(I_2(t)) = 1, \text{Var}(I_3(t)) = 10$.



(a) $t = 2$ (b) $t = 10$ (c) $t = 25$ (d) $t = 50$



(e) $t = 100$ (f) $t = 300$ (g) $t = 500$ (h) $t = 1000$

Figura 1.15: 2500 datos sobre $\mathbb{T}^2$ con coeficientes de perturbación mediante procesos de Poisson $N_1(t)$ y gaussiano inverso $I_2(t)$ con $\mathbb{E}(N_1(t)) = \mathbb{E}(I_2(t)) = 0.1$ y $\text{Var}(N_1(t)) = 0.1, \text{Var}(I_2(t)) = 10$.

En los ejemplos anteriores es posible observar los efectos de la media y la varianza de los procesos usados en la aproximación. Si la media es la misma y la varianza es pequeña

en cada uno de los procesos de la perturbación, la convergencia cuando el tiempo t se incrementa es rápida, como se aprecia en las Figuras 1.12 y 1.13. Por otro lado, si la media es la misma pero la varianza es grande en alguna de las perturbaciones, entonces la convergencia a la distribución uniforme en la variedad es más lenta concentrándose los puntos en mayor grado sobre esta entrada; como se aprecia en las Figuras 1.14 y 1.15.

En otros casos la convergencia es muy rápida. Esto se observa en las figuras de $\mathbb{S}^1$ y $\mathbb{S}^2$ cuando las entradas se perturban mediante procesos Poisson de parámetro $\lambda = 2$ pues la probabilidad de obtener ceros es muy pequeña. Este ejemplo y elecciones de distintas combinaciones de medias y varianzas chicas y grandes nos permiten ilustrar otros efectos que implican estas variaciones, mismos que se presentan en el Apéndice C.

Cuando los procesos Poisson tienen una media pequeña, el tiempo esperado para que salten es grande. Debido a ésto, en las perturbaciones se tienen puntos al centro de las variedades que podemos interpretar como *outliers*.

En el Capítulo 2 presentamos más ejemplos de simulación de nubes de puntos con estos mismos métodos incluyendo el cálculo de la homología y sus resúmenes topológicos.

1.2.3. Posibilidades de modelación

De acuerdo a lo observado en los ejemplos anteriores y los que se muestran en el Apéndice C se obtienen las siguientes conclusiones sobre posibilidades de modelación de nubes de puntos con la aproximación propuesta usando procesos de Lévy.

1. Si la media de las perturbaciones a la distribución uniforme es la misma y además la varianza es chica, la convergencia de las aproximaciones a la distribución se da de manera rápida.

2. La media juega un papel importante en la convergencia de las aproximaciones, cuando la media es grande e igual en cada caso, la convergencia a la distribución uniforme es muy rápida.

3. Si las entradas de la perturbación tienen media distinta, pero la varianza de una de ellas es "mucho más grande" que la varianza de la otra, la concentración de puntos se va a ver reflejada sobre la entrada que tenga mayor variabilidad. En este caso, la convergencia nunca se dará hacia la distribución uniforme en la variedad, pues no se satisfacen las hipótesis en la Proposición 3(a). Tomando esto en cuenta, es posible llevar a cabo simulaciones de nubes de datos que tengan concentraciones en algunas regiones de interés.

4. Si la media de las perturbaciones es la misma y la varianza es distinta, se verá una mayor concentración sobre la variable perturbada con mayor variabilidad. Aún con esto, la convergencia se da de manera lenta a la distribución uniforme pues si se satisfacen los supuestos en 3(a).

5. Esta forma de simular nubes de datos permite flexibilidad de modelación de diversas características de las nubes, algunas de ellas cercanas a la distribución uniforme.

6. Si bien las perturbaciones propuestas se construyen en base a procesos de Lévy, pensamos que no existe mucha diferencia si se usan otros procesos. Los procesos de Lévy ofrecen muchas posibilidades de modelación, y son fáciles de simular.

2

Preliminares de Homología Persistente

Este capítulo está destinado a introducir diversos conceptos y resultados tanto topológicos como estadísticos que se utilizarán a lo largo del presente trabajo de tesis.

En la Sección 2.1 mencionamos conceptos básicos de álgebra tales como clases de equivalencia y grupos cociente. En la Sección 2.2 describimos los conceptos topológicos que son clave para el desarrollo de la homología persistente. En la Sección 2.3 se detalla el fundamento teórico de la homología persistente, la cual es uno de los engranes de la maquinaria utilizada en el desarrollo del Análisis Topológico de Datos, incluyendo los complejos de Čech y Vietoris-Rips (VR). En la Sección 2.4 se presentan dos de los resúmenes de persistencia más usados: diagramas de persistencia y códigos de barra. En la Sección 2.5 se incluye la construcción del complejo de Morse-Smale (MS) como una primer alternativa a los complejos de Čech y Vietoris-Rips. Para el caso de los complejos de Morse-Smale se menciona cómo construir bandas de confianza en los diagramas de persistencia que ayudan a distinguir entre ruido topológico y características reales usando estimación de densidades vía el método de kernel. Finalmente, en la Sección 2.6 incluimos numerosos ejemplos de nubes de datos simuladas (con los métodos del Capítulo 1) y el cálculo de su homología con sus correspondientes resúmenes topológicos, comparando la efectividad de los complejos VR y MS.

2.1. Grupos cociente

En esta sección daremos definición a una clase particular de grupos abelianos, los grupos cociente. Este concepto es necesario para poder entender cómo podemos distinguir en clases de equivalencia agujeros *verdaderos* de los que no lo son (se explicará esto a mayor detalle en la Sección 2.3).

Definición 1. Decimos que la relación binaria $\sim$ sobre A es una *relación de equivalencia* sobre A si para cualesquiera $a, b, c \in A$ se satisfacen las siguientes propiedades:

1. $a \sim a$ (reflexiva),
2. Si $a \sim b$ entonces $b \sim a$ (simétrica),
3. Si $a \sim b$ y $b \sim c$ entonces $a \sim c$ (transitiva).

En base a una relación de equivalencia $\sim$ dada, podemos agrupar los elementos de un conjunto en subconjuntos disjuntos.

Definición 2. Si A es un conjunto y $\sim$ es una relación de equivalencia sobre A, entonces la *clase de equivalencia de* $a \in A$ es el conjunto $\{x \in A | a \sim x\}$. Denotamos la clase de equivalencia de $a \in A$ como $[a]$.

Lema 4. *Si* $[a]$ *y* $[b]$ *son dos clases de equivalencia en* A *entonces* $[a] \cap [b] = \emptyset$ *o* $[a] = [b]$.

Si tenemos una colección de subconjuntos disjuntos de A tales que su unión es A, decimos que tenemos una *partición de* A. Recíprocamente, si A tiene una partición podemos definir una relación de equivalencia inducida por la partición misma mediante la siguiente regla

$$a \sim b \Leftrightarrow a \text{ y } b \text{ están en el mismo elemento de la partición.}$$

Definición 3. Decimos que un conjunto de elementos G no vacío forma un *grupo* si en él se encuentra definida una operación binaria, llamada producto y denotada por $*$ tal que

1. $a, b \in G$ implica que $a * b \in G$ (cerradura).
2. $a, b, c \in G$ implica que $a * (b * c) = (a * b) * c$ (Ley asociativa).
3. Existe un elemento $e \in G$ tal que $a * e = e * a = a$ para todo $a \in G$ (existencia de un elemento identidad en G).
4. Para todo $a \in G$ existe un elemento $a^{-1} \in G$ tal que $a * a^{-1} = a^{-1} * a = e$ (existencia de inversos en G).

   Si además la operación producto en G satisface que

5. Para cualesquiera $a, b \in G$ se tiene que $a * b = b * a$,

   G se llama *grupo abeliano*.

Un ejemplo muy importante en el ATD es el grupo $\mathbb{Z}_2$ que consta de los enteros módulo 2, esto es: si $m,\ n \in \mathbb{Z}$ definimos la siguiente relación de equivalencia

$$m \sim n \Leftrightarrow m - n = 2k \qquad \text{con } k \in \mathbb{Z}.$$

Este grupo sólo consta de las clases de equivalencia [0] y [1], las cuales serán utilizadas como coeficientes en una construcción particular entre simplejos.

Definición 4. Un subconjunto no vacío H del grupo G es un subgrupo de G si y sólo si

1. $a, b \in H$ implica que $a * b \in H$,
2. $a \in H$ implica que $a^{-1} \in H$.

Usaremos la notación $H < G$ para indicar que H es subgrupo de G.

Definición 5. Sea G un grupo y $H < G$. Defínase la siguiente relación de equivalencia en G:

$$g \sim g' \Leftrightarrow g - g' \in H.$$

Al conjunto de clases de equivalencia de $\sim$ lo denotaremos como G/H. Definimos además la operación producto entre clases de equivalencia como la suma entre sus representantes, esto es:

$$[g] + [g'] = [g + g'].$$

Así, $(G/H, +)$ es un grupo abeliano, el cual es llamado *grupo cociente*.

Para mayor referencia acerca de grupos y ejemplos remitimos al lector al libro de Herstein (1988).

2.2. Conceptos topológicos

Es necesario definir una serie de conceptos básicos que nos ayuden a definir la homología, pues es la herramienta clave en el ATD. En esta sección se describen conceptos tales como topología, continuidad, simplejos, complejos simpliciales y transformaciones que podemos aplicar a éstos.

2.2.1. Espacios topológicos

Es posible abstraer la definición de lo que conocemos como un conjunto abierto y esto se da en base a la siguiente

Definición 6. Una topología sobre un conjunto X es una colección τ de subconjuntos de X, llamados *conjuntos abiertos*, que satisfacen:

1) Cualquier unión de elementos de τ pertenece a τ,

2) cualquier intersección finita de elementos de τ pertenece a τ,

3) $\emptyset$ y X pertenecen a τ.

Al par (X, τ) se le llama *espacio topológico*. Si no tenemos ningún problema para identificar la topología τ, simplemente decimos que "X es un espacio topológico".

A lo largo de este trabajo de tesis nos referiremos a un espacio topológico X bajo el entendido de que se trata de una dupla (X, τ) donde τ es alguna topología asociada. Se pueden ver ejemplos en Willard (1970).

Ejemplos de espacios topológicos sencillos son los espacios euclideanos $\mathbb{R}^d$, donde la topología estándar está conformada por productos de intervalos abiertos $(a_1, b_1) \times (a_2, b_2) \times (a_d, b_d)$ donde $a_i, b_i \in \mathbb{R} \cup \{-\infty, \infty\}$ para $i = 1, 2, ..., d$.

Si X es un espacio topológico, podemos referirnos a E como un conjunto cerrado en X si y sólo si su complemento E^c es abierto. En base a una colección de conjuntos cerrados tenemos el siguiente teorema.

Teorema 5. *Si $\mathcal{F}$ es la colección de todos los conjuntos cerrados de un espacio topológico X, entonces*

1) Cualquier intersección de miembros de $\mathcal{F}$ pertenece a $\mathcal{F}$.

2) Cualquier unión finita de elementos de $\mathcal{F}$ pertenece a $\mathcal{F}$.

3) X y $\emptyset$ pertenecen a $\mathcal{F}$.

Demostración. La prueba se basa en la aplicación de las leyes de De Morgan en conjunto con la definición de una topología. Se puede ver en Willard (1970). □

La topología también se puede caracterizar por medio de este teorema, esto es, utilizando a los conjuntos cerrados para definir la topología y a partir de esta construir resultados análogos a los que presentamos en este capítulo.

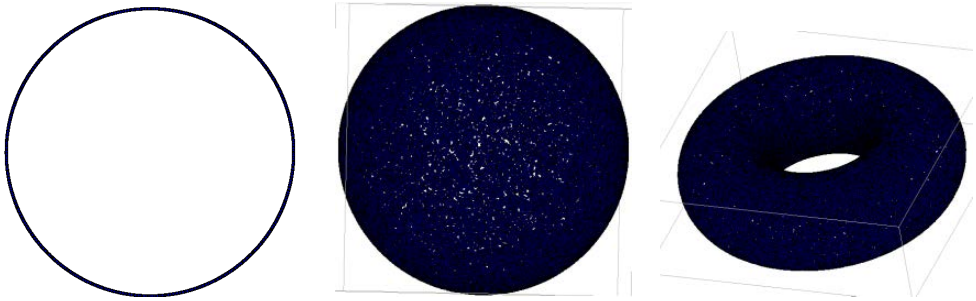
Un concepto básico que necesitaremos en el desarrollo de este trabajo es el de un subespacio topológico, pues resulta que las realizaciones de los complejos simpliciales (que describiremos más adelante) son subespacios topológicos de $\mathbb{R}^m$ para algún $m \geqslant 1$.

Definición 7. Sean (X, τ) es un espacio topológico, $Y \subset X$ un subconjunto de X. Entonces la colección

$$\tau_y = \{Y \cap U | U \in \tau\} \tag{2.1}$$

forma una topología para Y, y diremos que Y es un subespacio topológico de X con la topología heredada τ_Y.

Algunos ejemplos de subespacios topológicos son los que usamos como modelos en este trabajo y mencionamos en el Capítulo 1: el círculo $\mathbb{S}^1$ visto como subespacio de $\mathbb{R}^2$, la esfera $\mathbb{S}^2$ y el toro $\mathbb{T}^2$ vistos como subespacios de $\mathbb{R}^3$. En la Figura 2.1 mostramos un ejemplo de simulación en estas variedades.



(a) 10000 datos (b) 70000 datos (c) 70000 datos

Figura 2.1: $\mathbb{S}^1$, $\mathbb{S}^2$ y $\mathbb{T}^2$ de izquierda a derecha respectivamente.

El siguiente concepto es uno de los conocidos como axiomas de separabilidad que, como su nombre lo indica, trata de explicar la idea de "separación" entre cualquier par de puntos en un espacio topológico.

Definición 8. Sea X un espacio topológico. Diremos que X es un *espacio Hausdorff* si para todo par de elementos distintos $x_1, x_2 \in X$ existen abiertos U_1, U_2 de X para los cuales se satisface que $x_i \in U_i$, $i = 1, 2$ y $U_1 \cap U_2 = \emptyset$.

Podemos referirnos a un espacio Hausdorff como un espacio T_2, pues es el "código" con el que se identifica a este axioma de separabilidad. Existen los espacios $T_0, T_1, T_2, T_3, T_{3\frac{1}{2}}$ y T_4 para una mejor referencia véase el texto de Willard (1970).

También se define la *distancia de Hausdorff* entre dos espacios topológicos métricos X e Y como *"la distancia máxima de un conjunto al punto más cercano al otro conjunto"* (Rote, 1991). De manera analítica, dada una distancia d esta distancia se obtiene mediante la expresión

$$d_H(X, Y) = \max_{x \in X} \left\{ \min_{y \in Y} d(x, y) \right\}.$$

Definición 9. Una *cubierta de un espacio topológico* X es una colección $\{U_i\}_{i \in I}$ de conjuntos de X tales que $X = \bigcup_{i \in I} U_i$.

Decimos que $\{U_i\}_{i \in I}$ es una *cubierta abierta* si los conjuntos $U_i \in \tau$ para todo $i \in I$.

Definición 10. Sea X un espacio topológico, diremos que X es *compacto* si para cada cubierta abierta del espacio, existe una subcubierta finita.

Teorema 6 (Heine-Borel)**.** *Sea $X \subset \mathbb{R}^n$. X es compacto si y sólo si X es cerrado y acotado.*

2.2.2. Transformaciones y continuidad

Si tenemos una función $f : X \to Y$ entre dos espacios topológicos X y Y, podemos definir la noción de continuidad. Diremos que f es continua si se tiene que $f^{-1}(V) \in \tau_X$ para todo $V \in \tau_Y$.

Si $f : X \to Y$ es una función continua, diremos que f es un homeomorfismo si se satisfacen:

- f es biyectiva y
- $f^{-1} : Y \to X$ es continua.

  Además, decimos que X es homeomorfo a Y si existe un homeomorfismo entre X y Y

Definición 11. Sea $f : X \to Y$ una función continua e inyectiva, sea $Z = f(X)$ visto como un subespacio de Y, de manera que podemos hablar de la función $f' : X \to Z \subset Y$. Si $(f')^{-1}$ es continua, decimos que f es un *encaje topológico* (o simplemente *encaje*).

Un ejemplo de encaje es la función $f : [0, 2\pi) \to \mathbb{R}^2$ definida como $f(\theta) = (\cos\theta, \sin\theta)$.

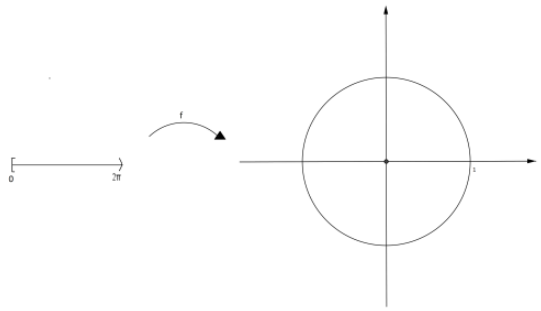


Figura 2.2: Encaje $f(\theta) = (\cos\theta, \sin\theta)$ de $[0, 2\pi)$ en $\mathbb{R}^2$.

Así como tenemos una definición de abierto y cerrado para conjuntos, se pueden extender estas definiciones para el caso de funciones. Decimos que una función es *abierta* si la imagen directa de un conjunto abierto es abierta; de manera análoga, decimos que una función es *cerrada* si la imagen directa de un conjunto cerrado es cerrada.

Algunas definiciones adicionales para funciones en espacios topológicos serían las siguientes:

- Dada una función sobre $f : X \to Y$, se dice que es una *función cociente* si se satisface que para todo abierto V en Y si y sólo si $f^{-1}(V)$ es abierto en X.
- Dos funciones continuas $f, g : X \to Y$ se dicen *homotópicas* si existe un mapeo continuo $H : X \times [0, 1] \to Y$ tal que $H(x, 0) = f(x)$ y $H(x, 1) = g(x)$.
- Decimos también, que una función $f : X \to Y$ es una *equivalencia homotópica* si existe una función $g : Y \to X$ tal que $g \circ f$ y $f \circ g$ sean homotópicas a las funciones identidad de X e Y respectivamente.
- Decimos que dos espacios X e Y son *homotópicamente equivalentes* si existe una equivalencia homotópica $f : X \to Y$.
- Un espacio X es *contráctil* si es homotópicamente equivalente al espacio que contiene un solo punto.

2.2.3. Complejos simpliciales

Una vez que hemos definido los conceptos algebraicos y topológicos básicos, podemos comenzar a establecer elementos que nos ayudarán a construir las componentes con las cuales poder inferir una estructura a un espacio a partir de una muestra dada de puntos, las piezas elementales por conveniencia son los simplejos.

Definición 12. Sea $V = \{v_0, v_1, ..., v_n\}$ un conjunto con $n + 1$ puntos en $\mathbb{R}^m$, se define la *envolvente convexa* como el conjunto $C(V) = \left\{ \sum_{i=0}^{n} \lambda_i v_i | \lambda_i \geqslant 0, \sum_{i=0}^{n} \lambda_i = 1 \right\}$.

Definición 13. Sean $v_0, v_1, ..., v_n$ puntos en $\mathbb{R}^m$ son tales que $v_0 - v_1, v_0 - v_2, ..., v_0 - v_n$ son linealmente independientes, decimos entonces que $v_0, v_1, ..., v_n$ son independientes.

Definición 14. Si $v_0, v_1, ..., v_n \in \mathbb{R}^m$ son independientes, definimos el *simplejo* generado por estos puntos como

$$\Delta(v_0, ..., v_n) = C(\{v_0, v_1, ..., v_n\}). \tag{2.2}$$

Al conjunto de puntos que generan al simplejo los llamaremos vértices. La dimensión de un simplejo generado por el conjunto de vértices $V = \{v_0, ..., v_n\}$ es $\#(V) - 1$. Si $\{e_0 = 0, e_1, ..., e_m\}$ es la base canónica de $\mathbb{R}^m$, decimos que $\Delta_m = \Delta(e_0, e_1, ..., e_m)$ es el *simplejo estándar de dimensión m.*

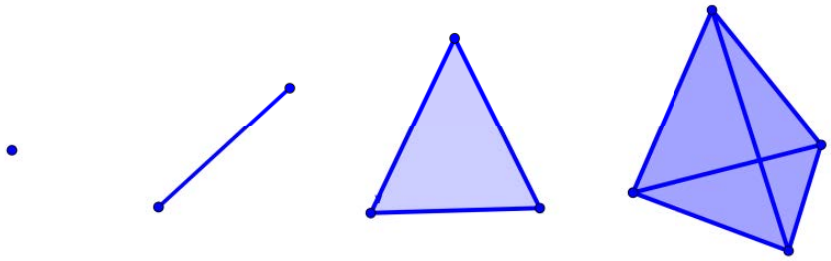


Figura 2.3: Simplejos estándar $\Delta_0, \Delta_1, \Delta_2$ y Δ_3

Utilizamos la noción de simplejo estándar pues se tiene la propiedad topológica de que todos los simplejos de dimensión n son homeomorfos. Así podemos caracterizar cualquier simplejo de dimensión n con su respectivo simplejo estándar.

Definición 15. Una función entre dos simplejos $f : \Delta(a_0, ..., a_n) \to \Delta(b_0, ..., b_m)$ es una *función simplicial* si para todo $i \in \{0, 1, ..., n\}$ se cumplen

i) $f(a_i) = b_j$ para algún $j \in \{0, 1, ..., m\}$,

ii) $f\left(\sum \lambda_i a_i\right) = \sum \lambda_i f(a_i)$.

Dada una función simplicial f, se satisfacen las siguientes propiedades:

1. Si f es continua en el simplejo, también es cerrada.
2. Si f es inyectiva, entonces es un encaje.
3. Si f es sobreyectiva, entonces es una función cociente.
4. Si f es biyectiva, entonces es un homeomorfismo.

Definición 16. Dados dos simplejos τ, σ en el mismo espacio euclideano decimos sin pérdida de generalidad que τ es cara (o subsimplejo) de σ si todos los vértices de τ son vértices de σ y lo denotamos $\tau \leqslant \sigma$. Si tenemos que $\dim \tau = n$ y $\tau \leqslant \sigma$ y además $\dim \sigma > n$ decimos que τ es una cara propia de σ, lo denotamos por medio de $\tau < \sigma$.

Un *complejo simplicial* $\mathcal{K}$ es un conjunto de simplejos que satisface

i) Si $\tau \leqslant \sigma$ y $\sigma \in \mathcal{K}$ entonces $\tau \in \mathcal{K}$,

ii) si $\tau, \sigma \in \mathcal{K}$, entonces $\tau \cap \sigma = \emptyset$ o $\tau \leqslant \sigma$ o $\sigma \leqslant \tau$.

Dado un complejo simplicial $\mathcal{K}$, definimos al *politopo* (o espacio subyacente) $|\mathcal{K}|$ como el subconjunto de $\mathbb{R}^n$ que es la unión de los simplejos de $\mathcal{K}$. La topología de $\mathcal{K}$ es la topología inducida en $|\mathcal{K}|$ por la topología estándar de $\mathbb{R}^n$.

2.3. Persistencia

La homología persistente es una de las herramientas utilizadas en la topología algebraica con la finalidad de encontrar características topológicas de los espacios de interés, tomando como elementos principales a los complejos simpliciales. Es a través de ella que podemos observar la evolución de tales características a medida que cambiamos un parámetro, de modo que podamos tener una estructura de multiresolución obteniendo así distintos niveles de detalle para cada valor de dicho parámetro. Hacemos referencia al libro de Edelsbrunner y Harer (2010) y a las notas de Biscay et al. (2016) para profundizar en la mayoría de los conceptos incluidos en esta sección

Definición 17. Dado un complejo simplicial $\mathcal{K}$ de dimensión n, definimos el *grupo de* $m-$*cadenas* (con la operación producto la suma usual) como el espacio vectorial libre

$$C_m(\mathcal{K}) = \{a_1\sigma_1 + \cdots + a_k\sigma_k | \sigma_i \text{ es un } m\text{-simplejo y } a_i \in \mathbb{Z}_2\} \qquad \text{para } 0 \leqslant m \leqslant n.$$

Si m no está en el rango entre 0 y n, diremos que $C_m(K) = \{0\}$. A los elementos de $C_m(\mathcal{K})$ los llamaremos $m-$cadenas.

Nota: *Cuando hablamos de espacios vectoriales libres nos referimos a lo siguiente: Si P es un conjunto finito y F es un campo. Definimos al conjunto $F[P]$ como el conjunto de todas las sumas formales de la forma $\sum_{p\in P} f_p p$, donde $f_p \in F$ para cada $p \in P$. Se define la* suma *de dos sumas formales como:*

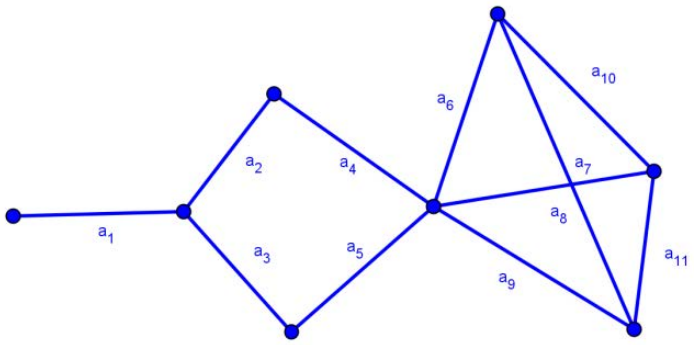
$$\left(\sum_{p\in P} f_p p\right) + \left(\sum_{p\in P} g_p p\right) = \sum_{p\in P} (f_p + g_p)\, p.$$

Definimos el producto escalar de alguna $f \in F$ y una suma formal como:

$$f \sum_{p\in P} f_p p = \sum_{p\in P} (f f_p) p.$$

Al tratarse de un espacio con coeficientes en $\mathbb{Z}_2$ (0 y 1), tenemos una manera geométrica bastante intuitiva de interpretar un elemento de $C_m(\mathcal{K})$. Imaginemos que tenemos un tablero con tubos de neón, donde cada uno es un $m-$simplejo; si el coeficiente es 1, indica que nuestro $m-$simplejo aparece encendido en la $m-$cadena, caso contrario si el coeficiente es 0. Por simplicidad, denotemos a los grupos de $m-$cadenas como C_m.

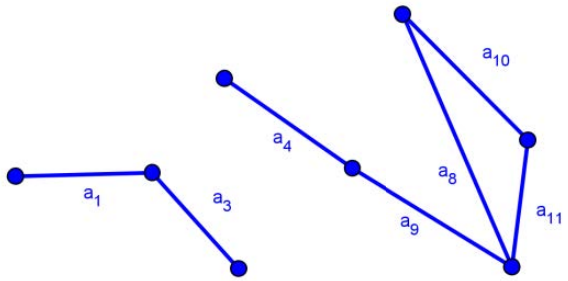
Ejemplo 1. Si tenemos el siguiente complejo simplicial $\mathcal{K}$ de dimensión 1,



la expresión como $1-$cadena es la siguiente

$$a_1 + a_2 + a_3 + a_4 + a_5 + a_6 + a_7 + a_8 + a_9 + a_{10} + a_{11}.$$

Pero en cambio, si "apagamos" algunas aristas y queda como



entonces la $1-$cadena que lo representa sería

$$a_1 + a_3 + a_4 + a_8 + a_9 + a_{10} + a_{11}.$$

Un complejo simplicial como el del Ejemplo 1 que acabamos de mostrar se puede considerar un $1-$ esqueleto de algún complejo simplicial que puede contener simplejos de dimensión mayor a 1. En otras palabras, un j esqueleto $S_j(\mathcal{K})$ de $\mathcal{K}$ consta de todos los simplejos de dimensión a lo más j.

Si denotamos $\sigma = [a_0 a_1 \cdots a_{m+1}]$ a un $m+1-$simplejo, se define un mapeo para σ como:

$$\delta_{m+1}\sigma = \sum_{j=0}^{m+1} [a_0 a_1 \cdots \hat{a}_j \cdots a_{m+1}],$$

donde $\hat{a}_j$ indica que se remueve ese vértice. A δ_{m+1} se le llama $m+1$-*ésimo mapeo frontera*. Una manera intuitiva de explicar el mapeo frontera es tomar un simplejo de dimensión

$m+1$ y mapearlo a la suma de todos sus caras de dimensión m. Se puede extender esta definición para $\delta_{m+1} : C_{m+1} \longrightarrow C_m$ para los complejos de cadenas de una manera natural, haciendo uso de linealidad del mapeo frontera de la siguiente manera:

$$\delta_{m+1}\left(\sum_{i=1}^{r} a_i\sigma_i\right) = \sum_{i=1}^{r} a_i\delta_{m+1}(\sigma_i).$$

De este modo, para un complejo simplicial $\mathcal{K}$ de dimensión n, se tiene el siguiente esquema entre grupos de $m-$cadenas:

$$0 \longrightarrow C_n(\mathcal{K}) \xrightarrow{\delta_n} \cdots \xrightarrow{\delta_3} C_2(\mathcal{K}) \xrightarrow{\delta_2} C_1(\mathcal{K}) \xrightarrow{\delta_1} C_0(\mathcal{K}) \xrightarrow{\delta_0} 0$$

A continuación presentamos un lema fundamental para las funciones frontera.

Lema 7. *Dado un complejo simplicial $\mathcal{K}$, la composición de dos funciones frontera consecutivas satisface que*

$$\delta_{m-1} \circ \delta_m = 0 \qquad m \geqslant 1.$$

Aquí, el lector puede comprobar que al componer (multiplicar) dos funciones frontera consecutivas el resultado siempre es 0.

Descrito lo anterior, podemos tener el siguiente par de definiciones en base a la función frontera.

Definición 18. Las fronteras de dimensión m de $\mathcal{K}$ son el conjunto

$$B_m(\mathcal{K}) = \mathrm{Im}(\delta_{m+1}).$$

Definición 19. Los ciclos de dimensión m de $\mathcal{K}$ son el conjunto

$$Z_m(\mathcal{K}) = \mathrm{Ker}(\delta_m).$$

Utilizando estas definiciones y el lema anterior se tiene que $B_m(\mathcal{K}) \subset Z_m(\mathcal{K})$ para cada m. Podemos dar paso a la definición de un grupo de homología.

Definición 20. Si $\mathcal{K}$ es un complejo simplicial. Se define el $m-$ésimo grupo de homología como el grupo cociente:

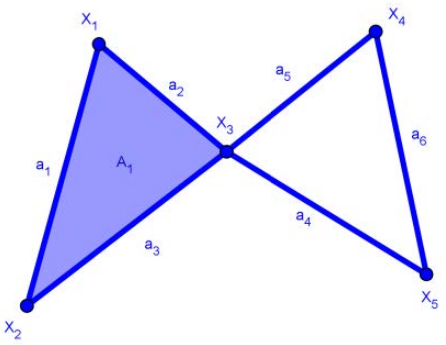
$$H_m(\mathcal{K}) = \frac{Z_m(\mathcal{K})}{B_m(\mathcal{K})}.$$

A la dimensión de $H_m(\mathcal{K})$ como espacio vectorial se le conoce como el $m-$ésimo número de Betti y se denota como $\beta_m(\mathcal{K})$.

Si queremos aproximar la estructura topológica de un espacio X a partir de una nube de puntos tomada de éste, un enfoque natural es mediante un complejo simplicial $\mathcal{K}$. De este modo, si $\mathcal{K}$ es una buena aproximación de X, entonces podemos calcular una buena aproximación de sus números de Betti $\beta_k(X)$.

Nota: Si no existe confusión, podemos omitir el subíndice del mapeo frontera δ_m y simplemente escribirlo como δ.

Para ilustrar mediante un ejemplo los conceptos que acabamos de definir, imagínese que tenemos la siguiente realización $\mathcal{K}$ de un complejo simplicial:



Si tomamos el simplejo $a_4 + a_5 + a_6$ y aplicamos δ tenemos

$$\delta(a_4 + a_5 + a_6) = X_3 + X_5 + X_3 + X_4 + X_4 + X_5 = 0,$$

de donde podemos ver que se trata de un ciclo *verdadero* pues no es imagen de un simplejo de dimensión mayor, mientras que si tomamos el $2-$simplejo A_1 y aplicamos δ tenemos

$$\delta A_1 = a_1 + a_2 + a_3$$

y al aplicar por segunda vez δ

$$\delta\delta A_1 = \delta(a_1 + a_2 + a_3) = X_1 + X_2 + X_2 + X_3 + X_1 + X_3 = 0$$

como cabría esperar debido al Lema 3. En este caso particular, también nos da 0 pues $\delta(a_1 + a_2 + a_3) = 0$ pero $a_1 + a_2 + a_3$ es imagen de A_1.

Con este ejemplo podemos ver dos cosas, la primera es que la composición de dos mapeos frontera consecutivos es 0 y la segunda, nos ayuda a identificar ciclos *verdaderos* de los que provienen de un simplejo de dimensión mayor.

Observación: Recordemos que estamos utilizando coeficientes sobre $\mathbb{Z}_2$. En nuestro ejemplo tenemos que los grupos de homología son

$$H_0(\mathcal{K}, \mathbb{Z}_2) = \mathbb{Z}_2, \qquad H_1(\mathcal{K}, \mathbb{Z}_2) = \mathbb{Z}_2, \qquad H_k(\mathcal{K}, \mathbb{Z}_2) = 0 \text{ para } k \geqslant 2.$$

Esto traduce como los números de Betti $\beta_0 = 1,\ \beta_1 = 1,\ \beta_k = 0$ para $k \geqslant 2$.

Existe un caso particular de dos variedades que comparten los mismos números de Betti $\beta_0 = 1,\ \beta_1 = 2,\ \beta_2 = 1$; estos son el toro $\mathbb{T}^2$ y la botella de Klein K_2. Podemos hacer una diferencia en este tipo de casos, por ejemplo extendiendo el uso de los coeficientes a $\mathbb{Z}$. Los espacios de homología quedan como:

$$H_k(\mathbb{T}^2, \mathbb{Z}) = \begin{cases} \mathbb{Z} & \text{si } k = 0 \\ \mathbb{Z} \oplus \mathbb{Z} & \text{si } k = 1 \\ \mathbb{Z} & \text{si } k = 2 \end{cases}, H_k(K_2, \mathbb{Z}) = \begin{cases} \mathbb{Z} & \text{si } k = 0 \\ \mathbb{Z} \oplus \mathbb{Z}_2 & \text{si } k = 1 \\ 0 & \text{si } k = 2 \end{cases}.$$

La idea principal es construir complejos simpliciales que calculen la homología de un espacio subyacente X o al menos tengan una fuerte relación con él. Primero empezamos por definir el nervio de una cubierta abierta.

Definición 21. Sea X un espacio topológico y $\mathcal{U} = \{U_\alpha\}_{\alpha \in A}$ cualquier cubierta de X. El nervio de $\mathcal{U}$, denotado por $N(\mathcal{U})$ es un complejo simplicial abstracto con conjunto de vértices A y donde una familia $\{\alpha_0, ..., \alpha_k\}$ genera un $k-$simplejo si y sólo si $U_{\alpha_0} \cap \cdots \cap U_{\alpha_k} \neq \emptyset$.

Nota: Un *complejo simplicial abstracto* es una familia $\mathcal{S}$ tal que para cualquier $\sigma \subset \mathcal{S}$ y cualquier subconjunto no vacío $\tau \subset \sigma$ se tiene que $\tau \in \mathcal{S}$.

De aquí, se desprende el siguiente teorema.

Teorema 8 (Teorema del nervio (Borsuk, 1948))**.** *Supóngase que X y $\mathcal{U}$ son como descritos anteriormente, y supóngase también que la cubierta consiste de una cantidad numerable de conjuntos abiertos. Supóngase además que para todo $\emptyset \neq S \subseteq A$, se tiene que $\cap_{s \in S} U_s$ o es contráctil o es vacía. Entonces $|N(\mathcal{U})|$ es homotópicamente equivalente a X.*

La clave es encontrar cubiertas apropiadas. Si el espacio en cuestión es métrico, una cubierta natural es la generada por la familia $\mathcal{B}_\varepsilon(X) = \{B_\varepsilon(x)\}_{x \in X}$ para algún $\varepsilon > 0$. De manera más general, para cualquier subconjunto $V \subseteq X$ para el cual se satisfaga que $X = \cup_{v \in V} B_\varepsilon(v)$, es posible construir el nervio de la cubierta $\{B_\varepsilon(v)\}_{v \in V}$. Nos referiremos a esta construcción como el complejo de Čech adjunto a V y ε, lo denotaremos como $\check{\mathrm{C}}(V, \varepsilon)$. El problema es que esta construcción es computacionalmente costosa, pues requiere de almacenamiento de complejos simpliciales de dimensiones incluso más altas que el espacio ambiente de X, por lo que una forma de abordar este problema es mediante una alternativa a la construcción conocida como el complejo *Vietoris-Rips*.

Definición 22. Sea (X, d) un espacio métrico. El complejo Vietoris-Rips para X con parámetro ε, denotado por $\mathrm{VR}(X, \varepsilon)$, es el complejo simplicial cuyo conjunto de vértices es X, y donde $\{x_0, x_1, ..., x_k\}$ genera un $k-$simplejo si y sólo si $d(x_i, x_j) \leqslant \varepsilon$ para toda $0 \leqslant i, j \leqslant k$.

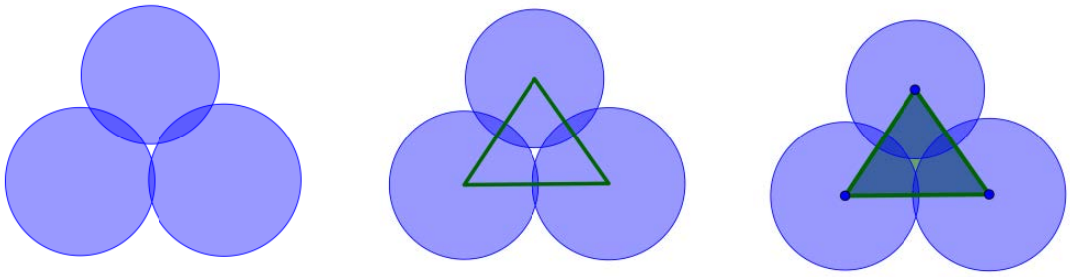


Figura 2.4: Comparativa entre los complejos de Čech (centro) y Vietoris-Rips (derecha).

La relación que existe entre los complejos de Čech y Vietoris-Rips está dada mediante la siguiente proposición.

Proposición 9. *Se satisfacen las siguientes inclusiones*

$$\check{C}(X, \varepsilon) \subseteq VR(X, 2\varepsilon) \subseteq \check{C}(X, 2\varepsilon)$$

Si aplicamos estas ideas a una nube de puntos discreta $P \subseteq X$, y variamos el valor de ε es posible ver cómo va cambiando el nervio de dicha cubierta (o en su defecto, el complejo de Vietoris-Rips). Esta noción se puede describir bajo el concepto de una filtración, que tiene la estructura de multiresolución que nos permite ver a diferentes "tiempos" cual es el comportamiento estructural de la nube de datos.

Definición 23. Sea $\mathcal{K}$ un complejo simplicial. Una *filtración* $\mathcal{F}$ de $\mathcal{K}$ es una colección de complejos simpliciales que cumplen

$$\emptyset = \mathcal{K}_0 \subset \mathcal{K}_1 \subset \mathcal{K}_2 \subset \cdots \subset \mathcal{K}_m = \mathcal{K},$$

donde $\mathcal{K}_i$ es subcomplejo simplicial de $\mathcal{K}_{i+1}$, $i = 0, ..., m-1$.

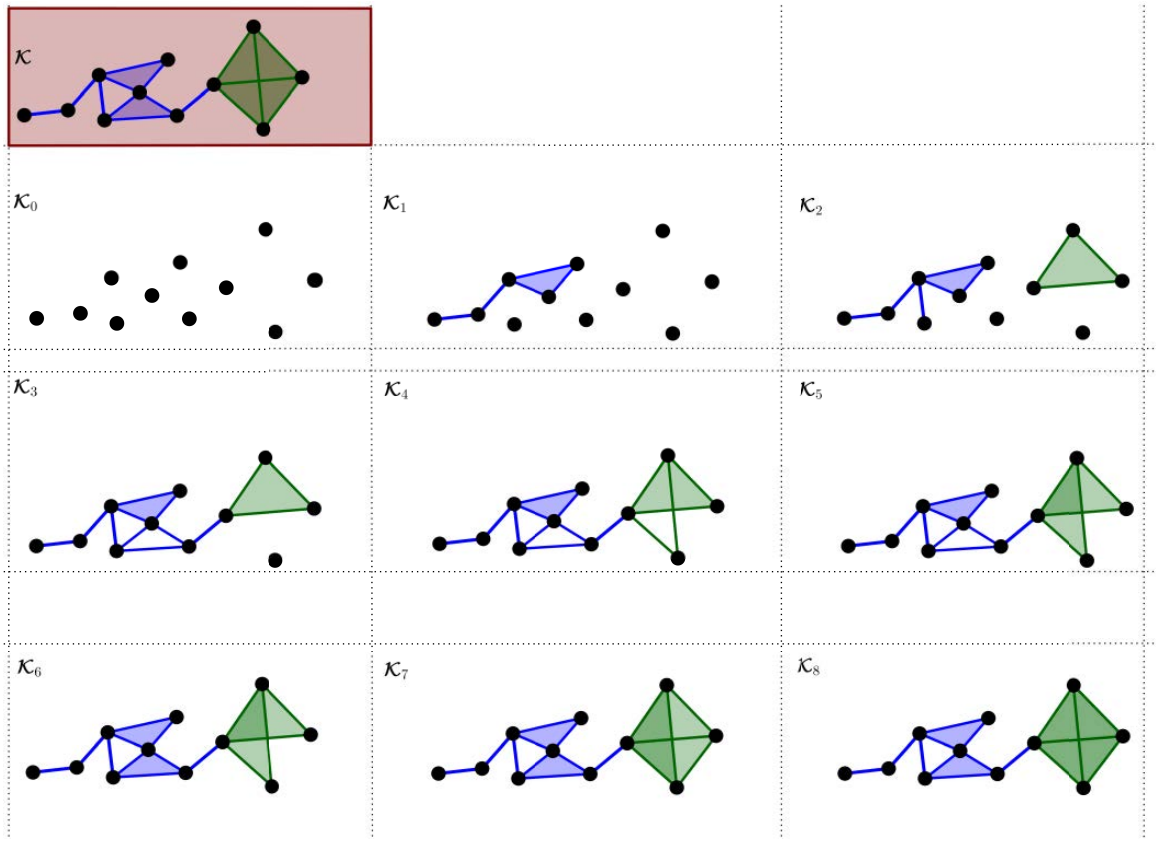


Figura 2.5: Ejemplo de una filtración.

Podemos así usar de manera natural la función inclusión $\iota_{i,i+1} : \mathcal{K}_i \hookrightarrow \mathcal{K}_{i+1}$ entre elementos de la filtración. Esta función inclusión induce un homomorfismo en los grupos de homología respectivos

$$(\iota_{ij}) :_* : H_p(\mathcal{K}_i) \longrightarrow H_p(\mathcal{K}_j).$$

Nota: Al hablar de la inclusión ι_{ij} nos referimos a la composición de inclusiones $\iota_j \circ \iota_{j-1} \circ \cdots \circ \iota_{i+1} \circ \iota_i$.

Definición 24. Sea $\mathcal{K}$ un complejo simplicial y $\mathcal{F}$ una filtración asociada. Definimos el $(i,j)-$*ésimo grupo de homología persistente de nivel* p como

$$H_{i,j;p}(\mathcal{K},\mathcal{F}) = \mathrm{Im}(\iota_{i,j})_*.$$

A la dimensión de este espacio vectorial lo denotamos por $\beta_{i,j;p}(\mathcal{K},\mathcal{F})$ y lo llamamos $(i,j)-$*ésimo número de Betti de nivel* p. Si se sobreentienden el valor de p, la filtración $\mathcal{F}$ y el complejo simplicial $\mathcal{K}$ del cual hablamos, simplemente escribiremos $H_{i,j}$ y $\beta_{i,j}$ respectivamente.

Definición 25. Decimos que la clase $\alpha \in H(\mathcal{K}_i)$ nace en el tiempo i si $\alpha \notin H_{i-1,i}$. Decimos también que i es el *tiempo de nacimiento* de α. Esto se ilustra en la siguiente figura:

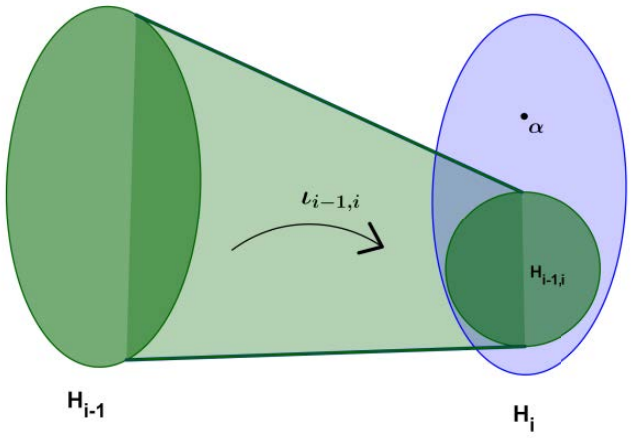


Figura 2.6: Esquema del nacimiento de una clase.

Definición 26. Decimos que la clase $\alpha \in H_i$ es ancestro de la clase $\beta \in H_j$ con $i \leqslant j$ si $\iota_{i,j}(\alpha) = \beta$. Presentamos la ilustración de este concepto en la siguiente figura:

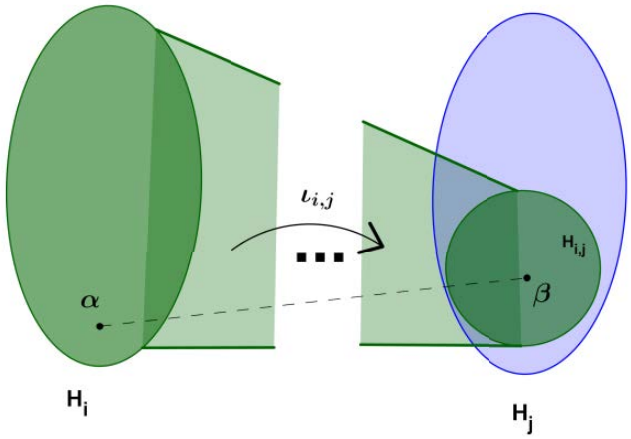


Figura 2.7: Esquema del ancestro de una clase.

Definición 27. Dada una clase $\alpha \in H_i$, nos referimos al *primer ancestro* de α como un ancestro de α cuyo tiempo de nacimiento es mínimo. Denotamos por $n(\alpha)$ al tiempo de nacimiento de los primeros ancestros de α. Por convención, el cero nace en $-\infty$, es decir $n(0) = -\infty$.

Definición 28. Decimos que una clase $\alpha \in H_j$ que nace en el tiempo i, muere en el tiempo $j+1$ si

$$\iota_{i,j}(\alpha) \notin H_{i-1,j} \text{ y } \iota_{i,j+1}(\alpha) \in H_{i-1,j+1}.$$

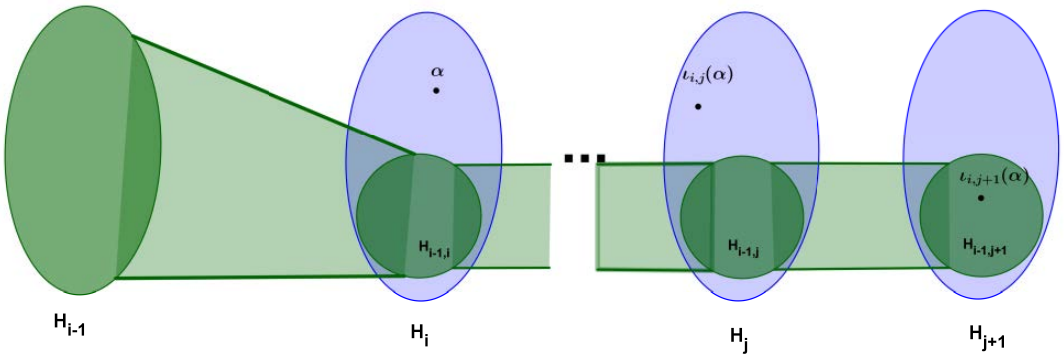


Figura 2.8: Esquema de la muerte de una clase.

El índice de persistencia de α es $j+1-i$.

Definición 29. Decimos que $\beta \in H_j$ es descendiente de la clase $\alpha \in H_i$ para $i \leqslant j$ si $\iota_{i,j}(\alpha) = \beta$.

Al primer descendiente de una clase $\alpha \in H_i$ que muere, en caso de existir, lo llamaremos el *último descendiente de* α y a su tiempo de muerte lo denotaremos por $m(\alpha)$. Si tal descendiente no existe, escribimos $m(\alpha) = \infty$.

Definición 30. Sea $\alpha \in H_i$ una clase no cero. Definimos la persistencia (o tiempo de vida) de α como $\text{pers}(\alpha) = m(\alpha) - n(\alpha)$.

2.4. Resúmenes de persistencia

2.4.1. Diagramas de persistencia

La persistencia de las clases en un grupo de homología de dimensión p se puede resumir mediante un *diagrama de persistencia*, el cual es un multiconjunto de puntos en el plano extendido con un punto de multiplicidad $\mu_{i,j}$ por cada par de puntos (i,j). Para ilustrar esto, simulamos un círculo unitario con "ruido pequeño" y presentamos su diagrama de persitencia asociado.

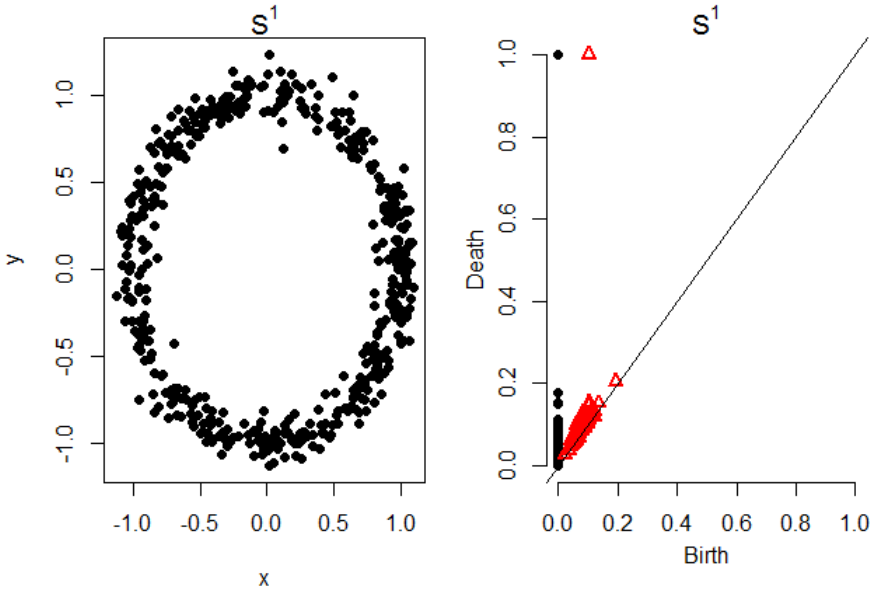


Figura 2.9: 400 puntos en S^1 y su diagrama de persistencia.

Nota: Con "ruido pequeño" nos referimos a una nube de puntos muestreada de la variable $M+\sigma N(0, I_n)$, donde M es una variable aleatoria en alguna variedad, y σ controla la varianza del ruido para el cual tomamos σ pequeño.

Podemos ver que el diagrama de persistencia muestra dos tipos de puntos, los negros que describen la persistencia de las clases homológicas de dimensión 0 (componentes conexas) y los rojos que describen la persistencia de aquellas de dimensión 1 (ciclos). Los puntos que se encuentran cerca de la diagonal pueden ser considerados como ruido, mientras que los que estén más alejados de ésta son los que podemos ver como características homológicas "reales". En este ejemplo, podemos ver que sólo persiste una componente conexa y un ciclo, es decir, sus números de Betti asociados son $\beta_0 = 1$ y $\beta_1 = 1$ que son característicos de una circunferencia.

Un estudio sobre conjuntos de confianza para diagramas de persistencia se hace en la tesis de González Cucurachi (2016), así como una aplicación del ATD en nichos ecológicos. Se hace uso de técnicas de submuestreo y concentración de medida.

2.4.1.1. Bandas de confianza vía Bootstrap

A partir del método de remuestreo Bootstrap (véase Efron y Tibshirani (1994)), podemos generar conjuntos de confianza (no confundirlos con intervalos de confianza) para los diagramas de persistencia. El algoritmo a utilizar es el siguiente:

1. Dada una muestra $X = \{x_1, ..., x_n\}$, calcule el estimador de densidad $\widehat{p}_h$.

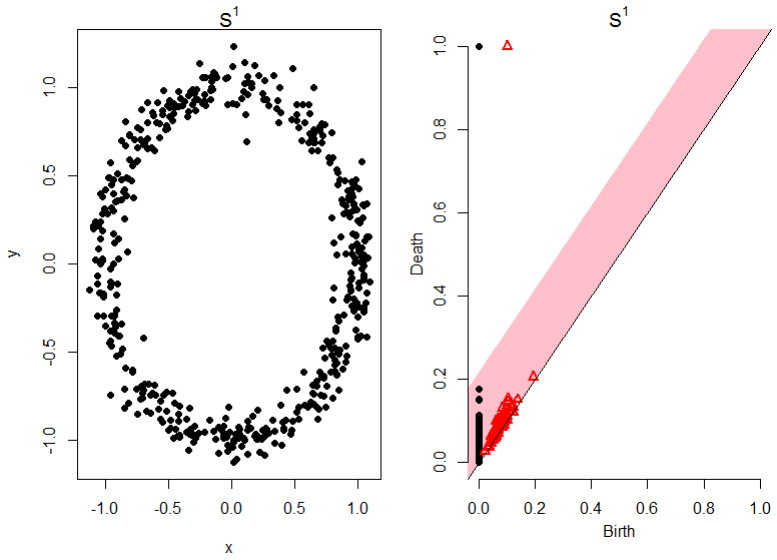
2. Tome una muestra con reemplazo $X^* = \{x_1^*, ..., x_n^*\}$ de $X = \{x_1, ..., x_n\}$ y calcule $\theta^* = \sqrt{n}\|\widehat{p}_h^*(x) - \widehat{p}_h(x)\|_\infty$, donde $\widehat{p}_h^*$ es el estimador de densidad calculado usando X^*.

3. Repita el paso anterior B veces para obtener $\theta_1^*, ..., \theta_B^*$.

4. Calcule $q_\alpha = \text{ínf}\left\{q : \frac{1}{B}\sum_{j=1}^{B} \mathbb{1}\left(\theta_j^* \geqslant q\right) \leqslant \alpha\right\}$

5. El conjunto de confianza de tamaño $1-\alpha$ para $\mathbb{E}\left[\widehat{p}_h\right]$ es $\left[\widehat{p}_h - \frac{q_\alpha}{\sqrt{n}}, \widehat{p}_h + \frac{q_\alpha}{\sqrt{n}}\right]$.

Para ilustrar, tomamos como ejemplo nuevamente el círculo unitario. La única diferencia que tiene con el que mostramos en la sección anterior, es que le añadimos una banda de confianza utilizando el método bootstrap.



La explicación de las características homológicas es la misma que anteriormente, en lo que nos ayuda la banda de confianza es a discriminar ruido topológico de características homológicas verdaderas presentas en la nube de puntos.

Así como definimos la distancia de Hausdorff entre dos espacios métricos, es posible definir una métrica entre dos diagramas de persistencia. En este caso, se define la *distancia cuello de botella* entre dos diagramas de persistencia D_1 y D_2 como

$$d_B(D_1, D_2) = \inf_{\gamma} \sup_{p \in D_1} \|p - h(p)\|;$$

donde h se encuentra en el conjunto de biyecciones entre los conjuntos D_1 y D_2.

A partir de este concepto y el de la distancia Hausdorff, se puede establecer un teorema de estabilidad para la filtración de Čech (ver Edelsbrunner y Morozov (2012)).

Teorema 10. *Sean D_1 y D_2 dos diagramas de persistencia obtenidos mediante alguna filtración para los espacios topológicos compactos $X, Y \subset \mathbb{R}^d$, entonces*

$$d_B(D_1, D_2) \leqslant d_H(X, Y).$$

2.4.2. Códigos de barra

Una alternativa a los diagramas de persistencia son los *códigos de barra*. Un código de barra es también una representación gráfica de la dimensión de los grupos de homología

H_p mediante segmentos de línea horizontales, el eje horizontal corresponde al parámetro de multiresolución, mientras que el eje vertical representa un orden (arbitrario) de los generadores de la homología. Las barras en un código de barras representan el tiempo de vida de las características topológicas a lo largo de la filtración, a través de ellas podemos medir la importancia de las características homológicas que se encuentren en una nube de puntos al ver cuanto tiempo logran "sobrevivir".

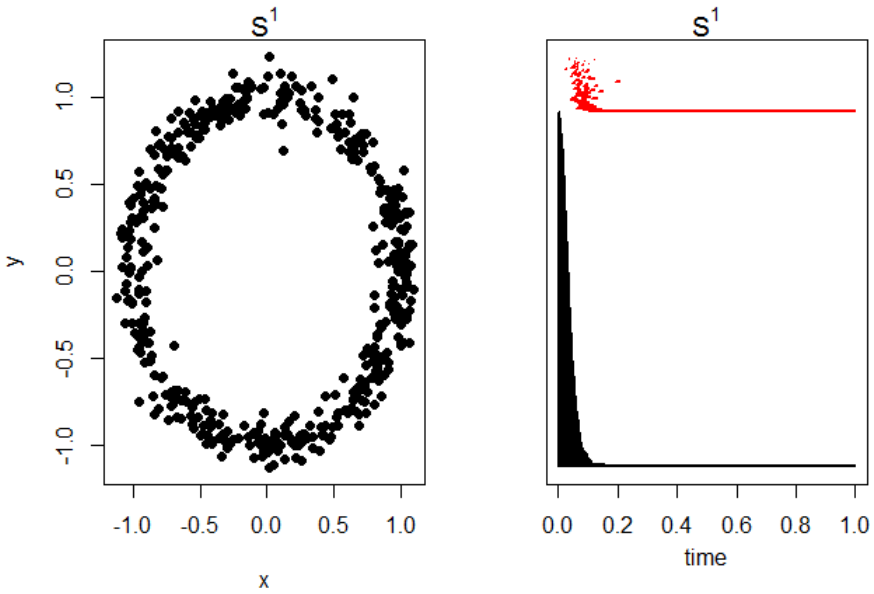


Figura 2.10: 400 puntos en S^1 y su código de barras.

Al igual que con el diagrama de persistencia, el código de barras está codificado mediante los colores negro para componentes conexas y rojo para ciclos. Aquí podemos ver con más detalle como hay un gran número de barras negras de tamaño muy pequeño, esto se debe a que al iniciar el cálculo de la persistencia se cuentan como componentes conexas todos los puntos de los que cuenta la nube D. Podemos ver cómo una barra negra persiste a lo largo de toda la filtración, de modo que determinamos que "realmente" hay una sola componente conexa. Un análisis similar nos dice que contamos con un solo ciclo.

En ocasiones sucede que para los dos resúmenes topológicos presentados es difícil hacer inferencia estadística, debido a que estos no se encuentran en un espacio vectorial, y conceptos como la media no tienen el sentido usual y es necesario considerar la media de Fréchet la cual no es necesariamente única. Existe otro tipo de resumen topológico, los panoramas de persistencia propuestos por Bubenik (2015), los cuales son funciones que pertenecen a un espacio vectorial para los cuales existen ley de grandes números y un teorema de límite central.

2.5. Filtración de Morse

La geometría y la topología de un espacio tienen una relación estrecha, si cambiamos un poco una la otra también se ve alterada. La teoría de Morse es una herramienta que nos

ayuda en el análisis de la estructura geométrica de un espacio X cuando esta se encuentra determinada por una función con un dominio "suave" (en este caso, una variedad). La teoría identifica los puntos críticos y a partir de ellos se genera un complejo simplicial mediante la descomposición de la variedad en regiones asociadas a los puntos críticos.

Hay una gran diversidad de funciones sobre una variedad que no se comportan de manera "bonita", una primera restricción sobre estas podría ser concentrarnos solamente sobre funciones continuas, pero en realidad esto no ayuda en mucho. Cuando hablamos de funciones *suaves* nos referimos a funciones para las cuales existen sus derivadas de cualquier orden. Un enfoque es mediante las funciones de Morse, que son funciones suaves sobre una variedad y además se caracterizan por tener sólo puntos críticos simples, esto es, si $h : \mathbb{M} \longrightarrow \mathbb{R}$ es una función real-valuada sobre una variedad $\mathbb{M}$ satisface

$$\nabla h(p) = 0,$$

o bien el flujo se estanca en p o se mueve hacia otro punto crítico. Llamamos a $h(p)$ el valor crítico de h en p. Cuando hablamos del flujo en un espacio vectorial nos referimos a una función que asigna un vector v_p en el espacio tangente para cada punto p en una variedad M.

La teoría de Morse clásica estudia los *conjuntos de subnivel* de funciones de Morse $h : M \longrightarrow \mathbb{R}$ sobre una variedad compacta

$$M_\alpha := h^{-1}((-\infty, \alpha]) = \{x \in M | h(x) \leqslant \alpha\} \subset M, \qquad \alpha \in (-\infty, \infty).$$

Podemos usar la monotonía de la función f para ver que los conjuntos de subnivel son anidados y por tanto podemos definir la persistencia a través de la sucesión de grupos de homología respectivos.

Teorema 11. *Si no existen valores críticos de h en el intervalo $(a, b]$ entonces M_a y M_b son homotópicamente equivalentes. Particularmente, tienen la misma homología.*

Por ejemplo, si $M = \mathbb{T}^2$ es el toro bidimensional y $h(x)$ la función altura en el punto $x \in M$. Cada valor real a tiene asociado un conjunto de subnivel asociado.

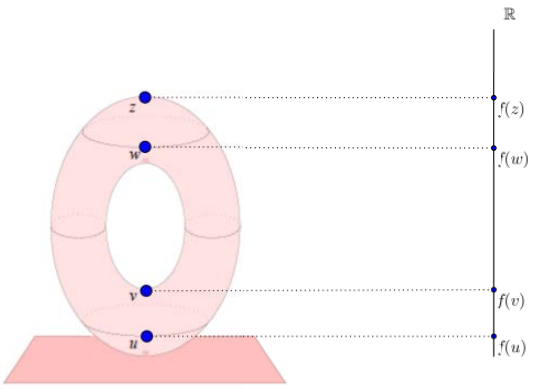


Figura 2.11: Función altura sobre el toro con puntos críticos u, v, w, z y conjuntos de nivel entre los distintos valores (Edelsbrunner y Harer (2010)).

Nos interesa ver el comportamiento de los conjuntos de subnivel conforme aumentamos los valores de a. Los eventos críticos suceden cuando a pasa los valores u, v, w y z en la Figura 2.11.

Para $a < h(u)$ el subconjunto de nivel es vacío, es decir $M_a = \emptyset$ y por tanto $H_m(M_a) \cong \{0\}$. Cuando $h(u) < a < h(v)$, podemos ver que el conjunto de subnivel asociado es un disco "doblado" el cual tiene el mismo tipo de homotopía que un punto, como se muestra en la Figura 2.12(a). Para $f(v) < a < f(w)$, el subconjunto de nivel es un cilindro, el cual tiene el mismo tipo de homotopía que un círculo, como se observa en la Figura 2.12(b). Para $f(w) < a < f(z)$ el conjunto de subnivel es un toro "tapado", y tiene el mismo tipo de homotopía que una figura en forma de 8: Finalmente, para $f(z) < a$, tenemos el toro completo que se obtiene de pegar un disco al toro "tapado". Esto es, terminamos la recuperación de la homología del toro.

Figura 2.12: Ilustración de la evolución de los conjuntos de nivel de la función altura sobre el toro $\mathbb{T}^2$ (Edelsbrunner y Harer (2010)).

Para llevar a cabo la construcción del complejo de Morse-Smale es necesario definir algunos conceptos adicionales.

Definición 31. Sea f una función de Morse, una variedad *estable* $S(p)$ o *inestable* $U(p)$ para un punto crítico p de f es:

$$S(p) = \{p\} \cup \{x \in \mathbb{M} | x \in \text{Im}(\gamma) \text{ y } dest(\gamma) = p\},$$
$$U(p) = \{p\} \cup \{x \in \mathbb{M} | x \in \text{Im}(\gamma) \text{ y } orig(\gamma) = p\}.$$

A γ se le conoce como *línea integral*, esta se trata de un camino máximal cuyos vectores tangente coincide con el gradiente de la función de Morse.

Definición 32. Una *función de Morse-Smale* es una función de Morse, $f : \mathbb{M} \to \mathbb{R}$ cuyas variedades estables e inestables se intersectan transversalmente.

Definición 33. Dos funciones $\sigma : \mathbb{R}^q \to \mathbb{M}$ y $\nu : \mathbb{R}^p \to \mathbb{M}$ se intersectan transversalmente en z si:

$$D\sigma_x(T\mathbb{R}^q_x) + D\nu_y(T\mathbb{R}^p_y) = T\mathbb{M}_z.$$

Asumiendo transversalidad, se construye el complejo Morse-Smale como sigue:

$\mathcal{S}_0$: Todos los puntos críticos se añaden como 0-variedades.

$\mathcal{S}_1$: Intervalos abiertos cuyos puntos finales son dos puntos del 0-esqueleto.

$\mathcal{S}_2$: Discos abiertos cuyas fronteras son un ciclo en el 1-esqueleto.

$\mathcal{S}_k$: Se continua esta construcción en dimensión mayor.

Ahora imaginemos que tenemos una nube de puntos $X = \{X_1, ..., X_n\}$ que provienen de una densidad $f(x)$ que es desconocida para nosotros. Podemos hacer uso de estimaciones de densidades para obtener un estimador $\hat{f}(x)$ de $f(x)$. Un caso particular es el estimador de densidades via kernel (o núcleo), el cual podemos considerar como un caso particular de las funciones suaves utilizadas en la teoría de Morse, este se define como sigue en el caso de dimensión 1.

Definición 34. Una función $K : \mathbb{R} \longrightarrow \mathbb{R}$ la llamaremos un *kernel (núcleo)* si es simétrica y satisface que $\int K = 1$. Dados $X_1, ..., X_n$ es una muestra aleatoria de densidad f sobre $\mathbb{R}$, un kernel K especificado y una constante h llamada ancho de banda, se define el *estimador por kernel de* f como

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right).$$

Nota: Al referirnos a una muestra aleatoria, estamos hablando de un conjunto de observaciones de variables aleatorias idénticamente distribuidas.

Por ejemplo, si tenemos datos simulados en S^1 con distribución uniforme, la estimación por Kernel de su densidad sería la siguiente:

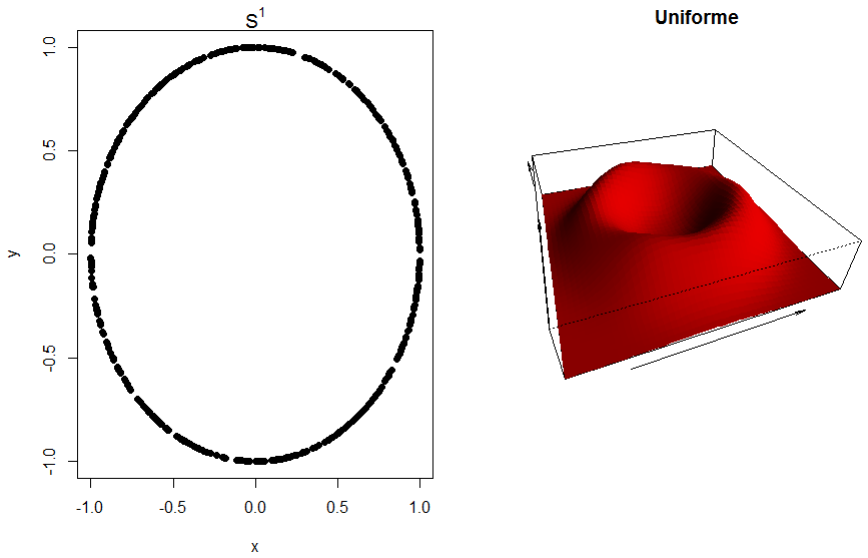


Figura 2.13: Simulación de 500 puntos en S^1 con distribución uniforme.

Si en cambio tenemos distribuciones con repulsión como es el caso de la distribución de los valores propios de una matriz GUE, tenemos el siguiente comportamiento mostrado en la Figura 2.14.

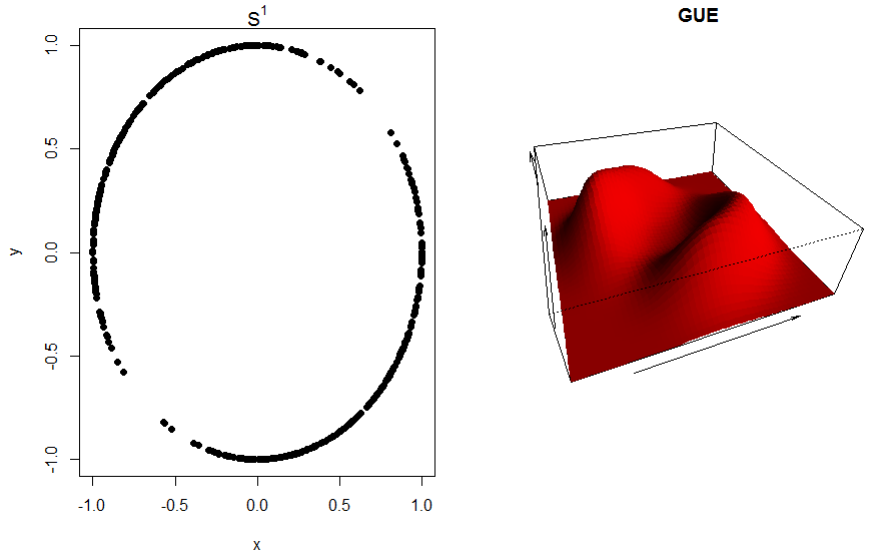


Figura 2.14: Simulación de 500 puntos en S^1 con distribución cociente GUE.

O podríamos tener una distribución donde sus entradas estén altamente correlacionadas:

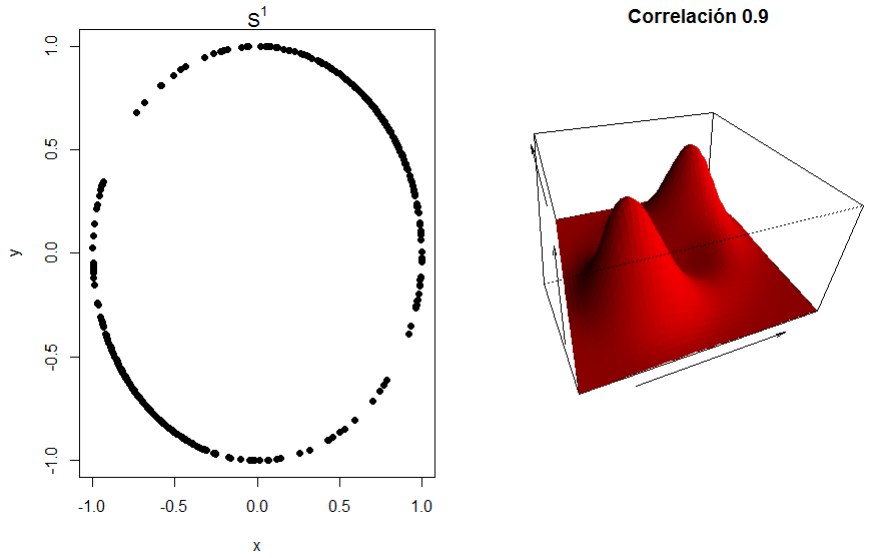


Figura 2.15: Simulación de 500 puntos en S^1 con correlación $\rho = 0.9$.

Resulta que el estimador vía kernel es consistente para f pues se satisface el siguiente

Teorema 12. *Sea $X_1, ..., X_n$ una muestra aleatoria con función de densidad $f(x)$ y $x \in \mathbb{R}$ fijo. Bajo supuestos adecuados (ver Tsybakov (2009)) se tiene que*

$$\hat{f}_h(x) \xrightarrow[n\to\infty, h\downarrow 0]{Pr} f(x),$$

donde $\xrightarrow{Pr}$ denota convergencia en probabilidad.

Se puede extender el concepto de estimadores vía kernel a un espacio multidimensional, para mayor información refiérase a (Biscay et al., 2016, Sección 5.4.5).

Una manera de medir la "diferencia" entre dos nubes de puntos en un espacio métrico es mediante la estimación de sus densidades respectivas y en base a esto calcular su distancia del supremo, esto es, dadas dos funciones f y g en el conjunto de funciones acotadas definidas en el espacio métrico X en cuestión, la *distancia del supremo (o distancia infinito)* se define como:

$$\|f - g\|_\infty := \sup_{x \in X} |f(x) - g(x)|. \tag{2.3}$$

2.6. Ejemplos: Distribución uniforme y aproximaciones

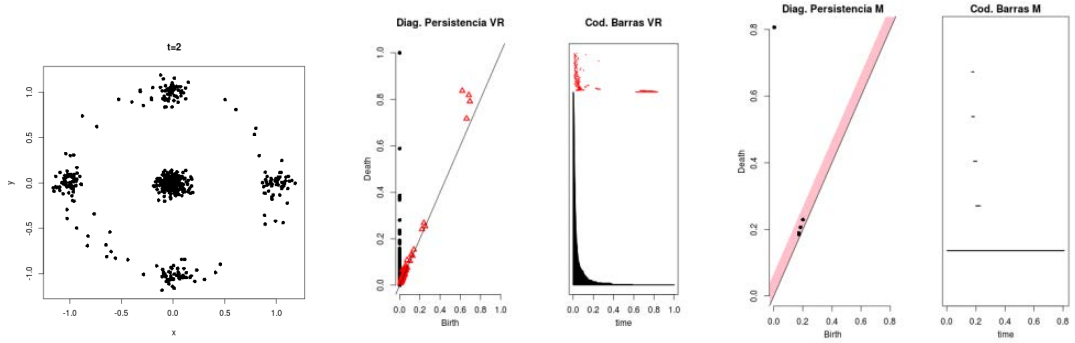
Presentamos algunos ejemplos de simulaciones que siguen la distribución uniforme así como aproximaciones a esta siguiendo el método ilustrado en el Capítulo 1, usando la perturbación con un Proceso de Poisson de parámetro $\lambda = 0.2$. Para los casos de nubes de datos en dimensiones 2 y 3 mostramos los datos, presentamos los diagramas de persistencia y códigos de barra para las filtraciones Vietoris-Rips y de Morse.

En la filtración VR fijamos el radio de la filtración dependiendo de la variedad con la que estemos trabajando pues existen casos en los que el algoritmo no puede calcular la homología debido al alto costo computacional. En el caso de la filtración de Morse lo que variamos es el ancho de banda h del estimador kernel y lo fino de la rejilla donde éste se evalúa. Teniendo el tamaño óptimo de ambos parámetros es posible ahorrar recursos computacionales y a su vez describir de manera más precisa la geometría subyacente de la nube de datos.

Como vimos en la Sección 1.2, cuando el tiempo que dejamos correr un proceso de Lévy es pequeño, la distribución de los datos tiende a concentrarse en la intersección de $\mathbb{S}^1$ con los ejes cartesianos. Conforme el tiempo del proceso avanza se parece a la distribución que nos da al hacer cociente con las entradas Cauchy(0,1) para después converger a la distribución uniforme.

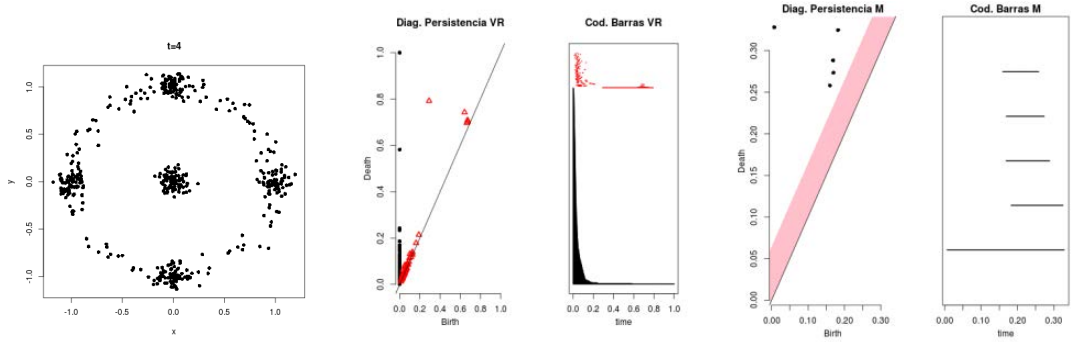
Los diagramas de persistencia que mostramos son realizados con la paquetería TDA de R (para detalles de su uso véase Fasy et al. (2014)). Los puntos negros indican agujeros de dimensión 0 (componentes conexas), los triángulos rojos indican agujeros de dimensión 1 (ciclos) y los círculos azules indican agujeros de dimensión 2 (huecos).

2.6.1. $\mathbb{S}^1$



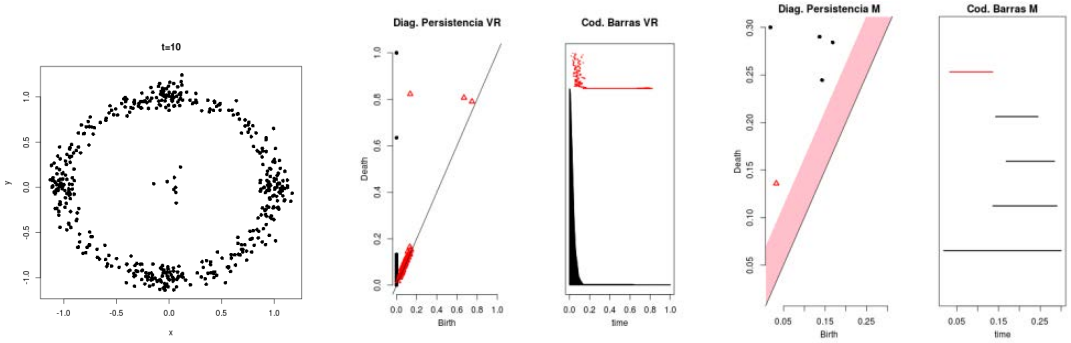
(a) 500 datos sobre $\mathbb{S}^1$ (b) Filtración VR (c) Filtración de Morse

Figura 2.16: 500 datos sobre $\mathbb{S}^1$ con coeficientes de pertubación mediante procesos de Poisson de parámetro $\lambda = 0.2$ para $t = 2$.



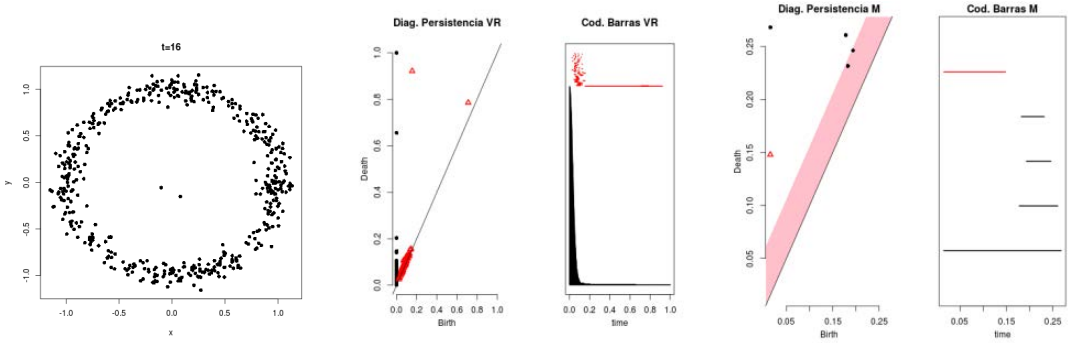
(a) 500 datos sobre $\mathbb{S}^1$ (b) Filtración VR (c) Filtración de Morse

Figura 2.17: Simulación sobre $\mathbb{S}^1$ con coeficientes de perturbación mediante procesos de Poisson de parámetro $\lambda = 0.2$ para $t = 4$.



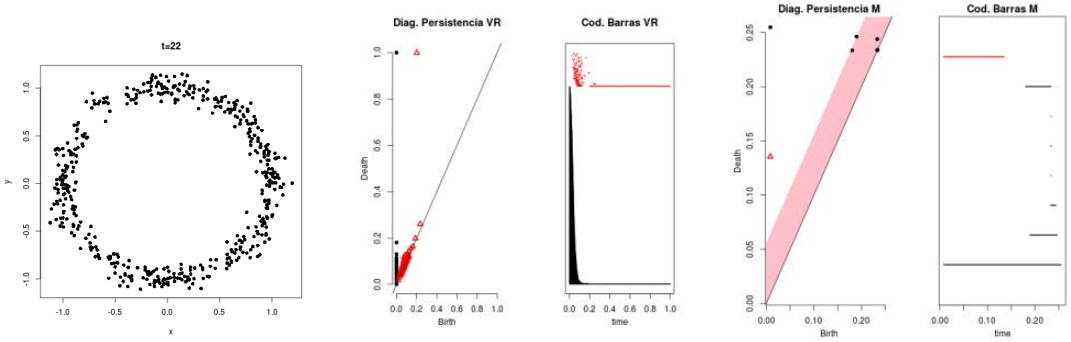
(a) 500 datos sobre $\mathbb{S}^1$ (b) Filtración VR (c) Filtración de Morse

Figura 2.18: Simulación sobre $\mathbb{S}^1$ con coeficientes de perturbación mediante procesos de Poisson de parámetro $\lambda = 0.2$ para $t = 10$.



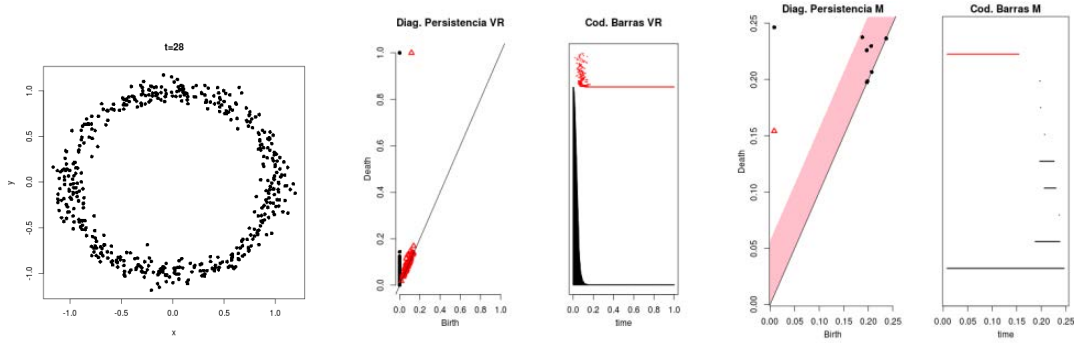
(a) 500 datos sobre $\mathbb{S}^1$ (b) Filtración VR (c) Filtración de Morse

Figura 2.19: Simulación sobre $\mathbb{S}^1$ con coeficientes de perturbación mediante procesos de Poisson de parámetro $\lambda = 0.2$ para $t = 16$.



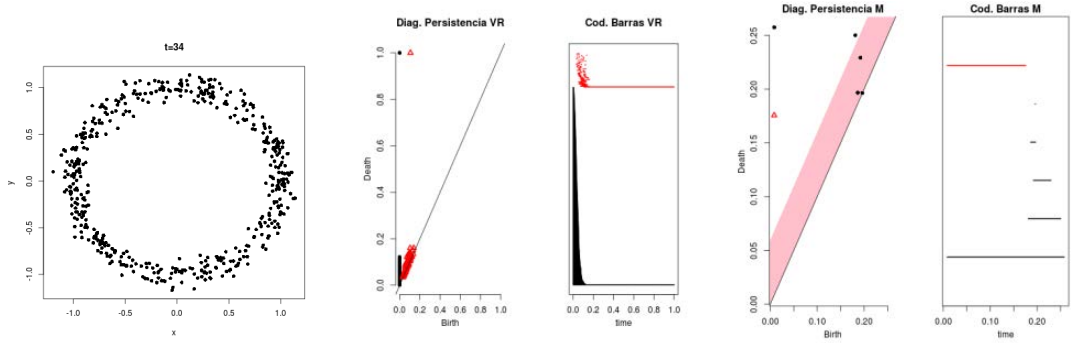
(a) 500 datos sobre $\mathbb{S}^1$ (b) Filtración VR (c) Filtración de Morse

Figura 2.20: Simulación sobre $\mathbb{S}^1$ con coeficientes de perturbación mediante procesos de Poisson de parámetro $\lambda = 0.2$ para $t = 22$.



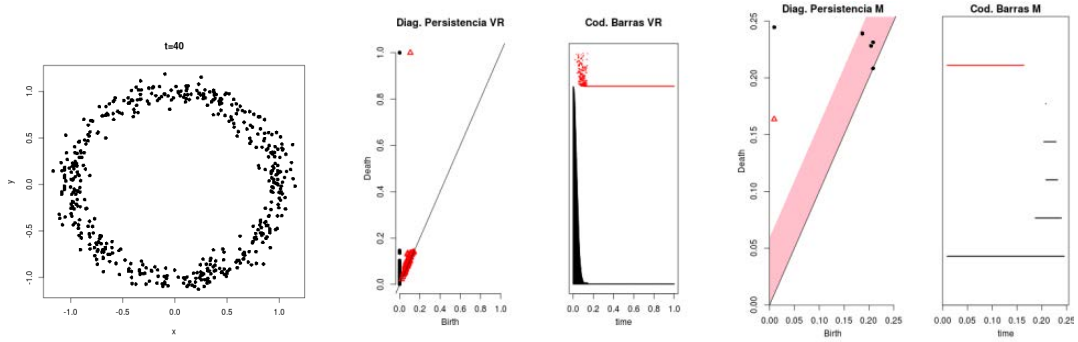
(a) 500 datos sobre $\mathbb{S}^1$ (b) Filtración VR (c) Filtración de Morse

Figura 2.21: Simulación sobre $\mathbb{S}^1$ con coeficientes de perturbación mediante procesos de Poisson de parámetro $\lambda = 0.2$ para $t = 28$.



(a) 500 datos sobre $\mathbb{S}^1$ (b) Filtración VR (c) Filtración de Morse

Figura 2.22: Simulación sobre $\mathbb{S}^1$ con coeficientes de perturbación mediante procesos de Poisson de parámetro $\lambda = 0.2$ para $t = 34$.



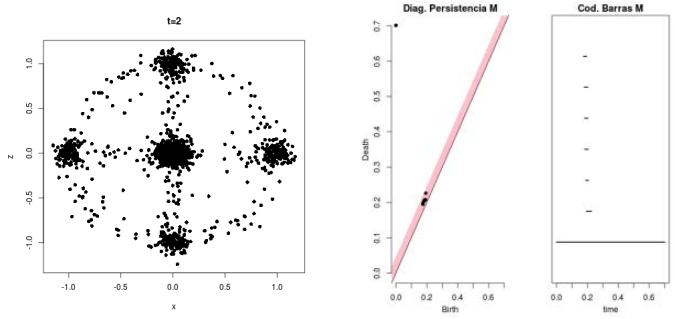
(a) 500 datos sobre $\mathbb{S}^1$ (b) Filtración VR (c) Filtración de Morse

Figura 2.23: Simulación sobre $\mathbb{S}^1$ con coeficientes de perturbación mediante procesos de Poisson de parámetro $\lambda = 0.2$ para $t = 40$.

De las Figuras 2.11-2.18 de $\mathbb{S}^1$ es posible observar que para los tiempos $t = 2, 4, 10, 16$ el algoritmo VR es capaz de detectar dos componentes conexas (el círculo y los puntos en el centro) pero no le es posible capturar la concentración de puntos que se tiene en los ejes cartesianos, situación que el algoritmo MS sí logra detectar. Sin embargo, MS no toma como un círculo a la forma subyacente de la nube de datos hasta el tiempo $t = 10$ que es cuando la densidad de puntos en los ejes cartesianos disminuye. A partir de $t = 16$, los dos algoritmos detectan una componente conexa y un agujero de dimensión 1, lo cual es característico de $\mathbb{S}^1$.

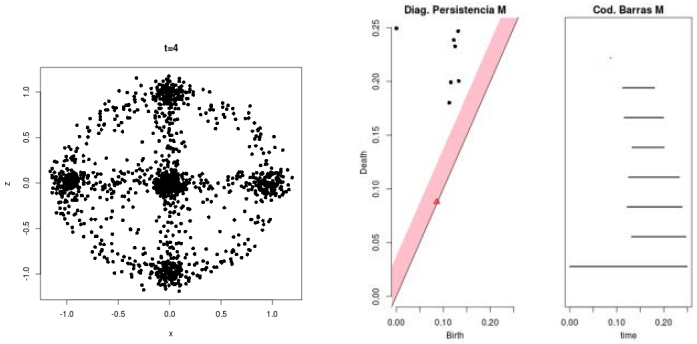
2.6.2. $\mathbb{S}^2$

También en el caso de una perturbación de la distribución uniforme mediante un proceso de Poisson de parámetro $\lambda = 0.2$, no fue posible calcular con el algoritmo VR la homología con un valor de filtración máximo de 0.8 para las aproximaciones en $\mathbb{S}^2$ para los tiempos $t = 2, 4$. Se redujo el valor de la filtración a 0.6 pero tampoco fue posible calcular la persistencia debido al uso alto de los recursos del servidor. Por ello no se muestran los diagramas de persistencia en el caso VR para los tiempos indicados para estos valores de filtración.



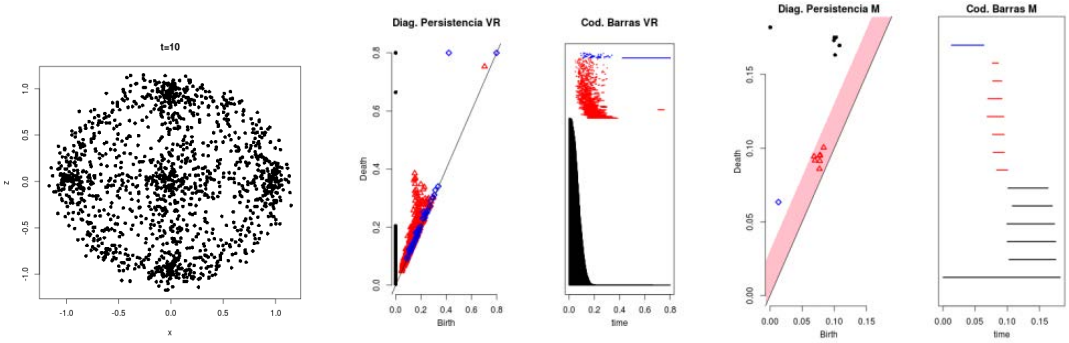
(a) 1500 datos sobre $\mathbb{S}^2$ (b) Filtración de Morse

Figura 2.24: Simulación sobre $\mathbb{S}^2$ con coeficientes de perturbación mediante procesos de Poisson de parámetro $\lambda = 0.2$ para $t = 2$.



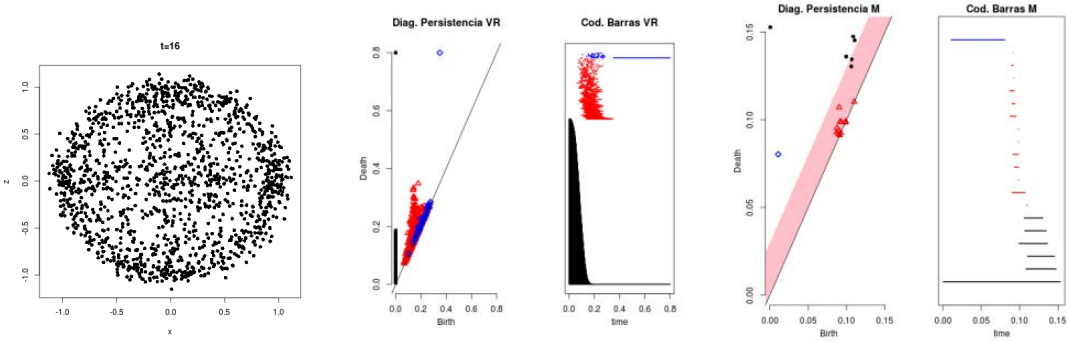
(a) 1500 datos sobre $\mathbb{S}^2$ (b) Filtración de Morse

Figura 2.25: Simulación sobre $\mathbb{S}^2$ con coeficientes de perturbación mediante procesos de Poisson de parámetro $\lambda = 0.2$ para $t = 4$.



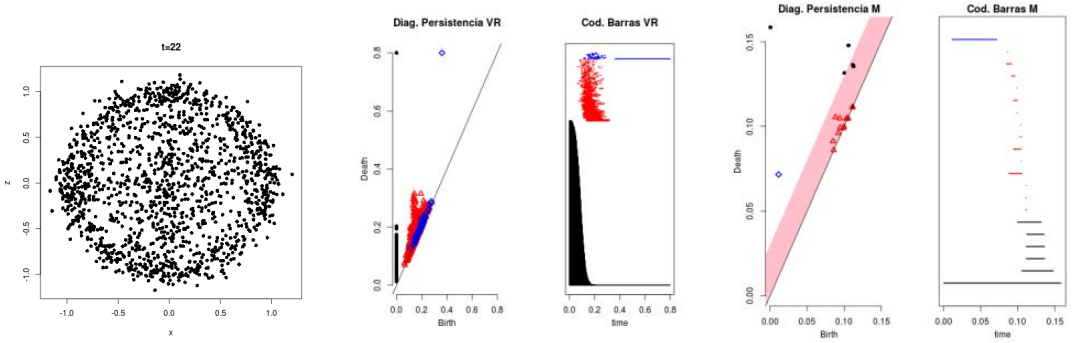
(a) 1500 datos sobre $\mathbb{S}^2$ (b) Filtración VR (c) Filtración de Morse

Figura 2.26: Simulación sobre $\mathbb{S}^2$ con coeficientes de perturbación mediante procesos de Poisson de parámetro $\lambda = 0.2$ para $t = 10$.



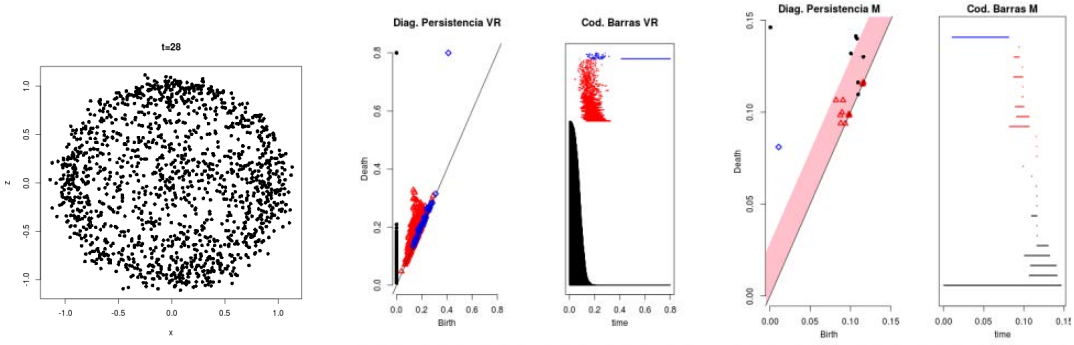
(a) 1500 datos sobre $\mathbb{S}^2$ (b) Filtración VR (c) Filtración de Morse

Figura 2.27: Simulación sobre $\mathbb{S}^2$ con coeficientes de perturbación mediante procesos de Poisson de parámetro $\lambda = 0.2$ para $t = 16$.



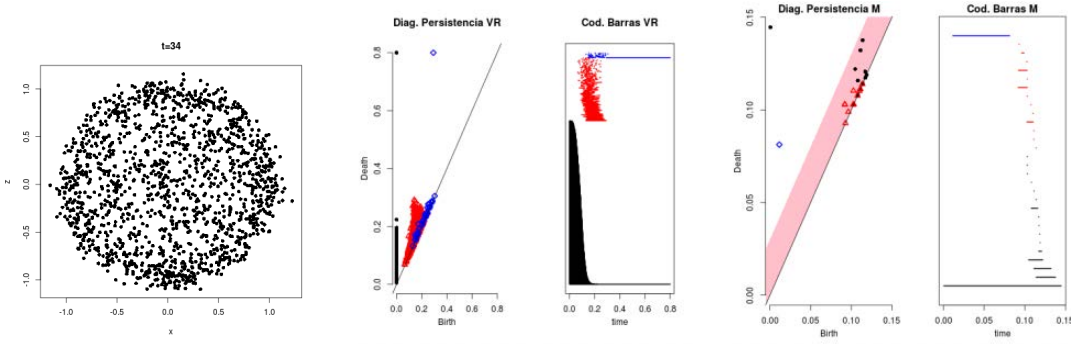
(a) 1500 datos sobre $\mathbb{S}^2$ (b) Filtración VR (c) Filtración de Morse

Figura 2.28: Simulación sobre $\mathbb{S}^2$ con coeficientes de perturbación mediante procesos de Poisson de parámetro $\lambda = 0.2$ para $t = 22$.



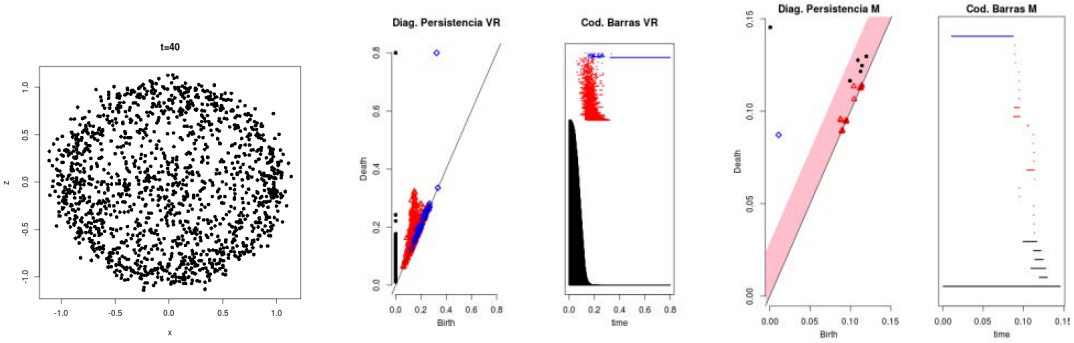
(a) 1500 datos sobre $\mathbb{S}^2$ (b) Filtración VR (c) Filtración de Morse

Figura 2.29: Simulación sobre $\mathbb{S}^2$ con coeficientes de perturbación mediante procesos de Poisson de parámetro $\lambda = 0.2$ para $t = 28$.



(a) 1500 datos sobre $\mathbb{S}^2$ (b) Filtración VR (c) Filtración de Morse

Figura 2.30: Simulación sobre $\mathbb{S}^2$ con coeficientes de perturbación mediante procesos de Poisson de parámetro $\lambda = 0.2$ para $t = 34$.



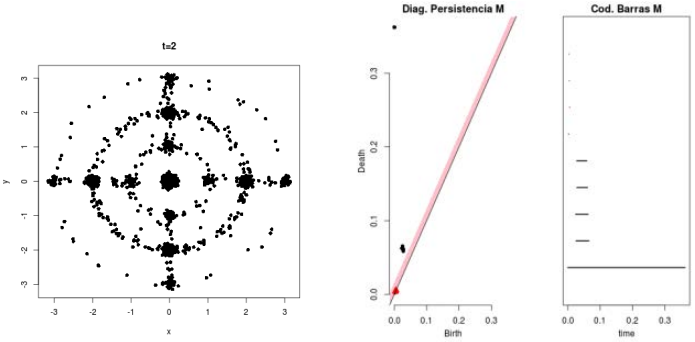
(a) 1500 datos sobre $\mathbb{S}^2$ (b) Filtración VR (c) Filtración de Morse

Figura 2.31: Simulación sobre $\mathbb{S}^2$ con coeficientes de perturbación mediante procesos de Poisson de parámetro $\lambda = 0.2$ para $t = 40$.

En las Figuras 2.19–2.26 se puede observar que para el tiempo $t = 10$ el algorimo VR detecta dos componentes conexas (la esfera y los puntos al centro) así como el agujero 2-dimensional. Es a partir de este tiempo que el algoritmo detecta los números de Betti característicos de la esfera $\mathbb{S}^2(\beta_0 = 1,\ \beta_1 = 0,\ \beta_2 = 1)$. En el caso del algoritmo MS podemos ver que se detectan las concentraciones de puntos en los ejes cartesianos hasta el tiempo $t = 28$; a partir de aquí se encuentran los números de Betti de la esfera $\mathbb{S}^2$, sin embargo, el agujero 2−dimensional se detecta desde $t = 10$.

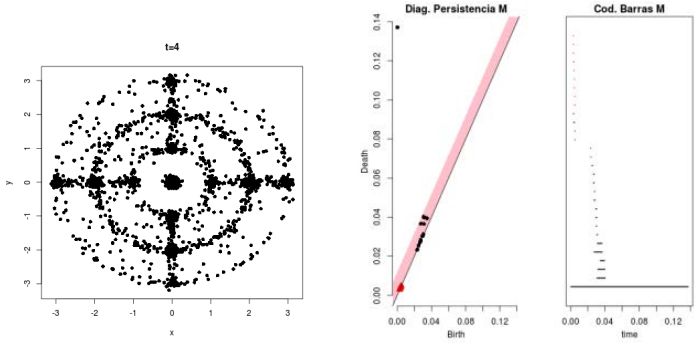
2.6.3. $\mathbb{T}^2$

Al igual que con $\mathbb{S}^2$, el algoritmo VR no es capaz de calcular la homología de las perturbaciones de la distribución uniforme en $\mathbb{T}^2$ mediante procesos de Poisson, para ningún valor de filtración relevante para los tiempos $t = 2, 4$. Se hace uso completo de los recursos del servidor en estos casos.



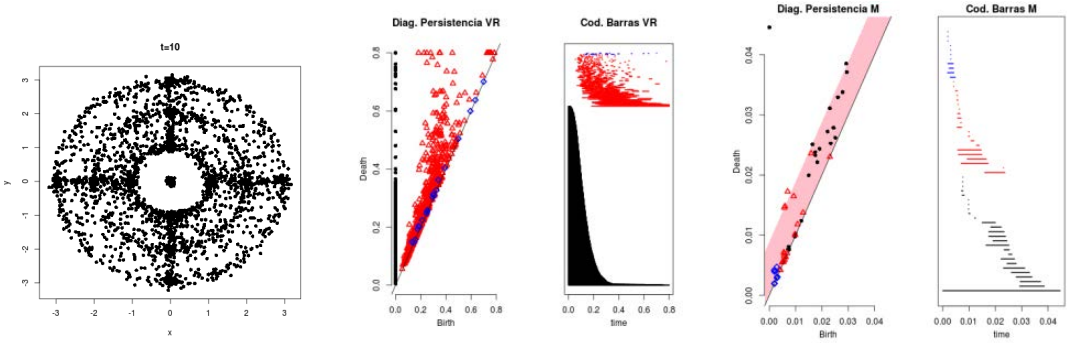
(a) 2500 datos sobre $\mathbb{T}^2$ (b) Filtración de Morse

Figura 2.32: Simulación sobre $\mathbb{T}^2$ con coeficientes de perturbación mediante procesos de Poisson de parámetro $\lambda = 0.2$ para $t = 2$.



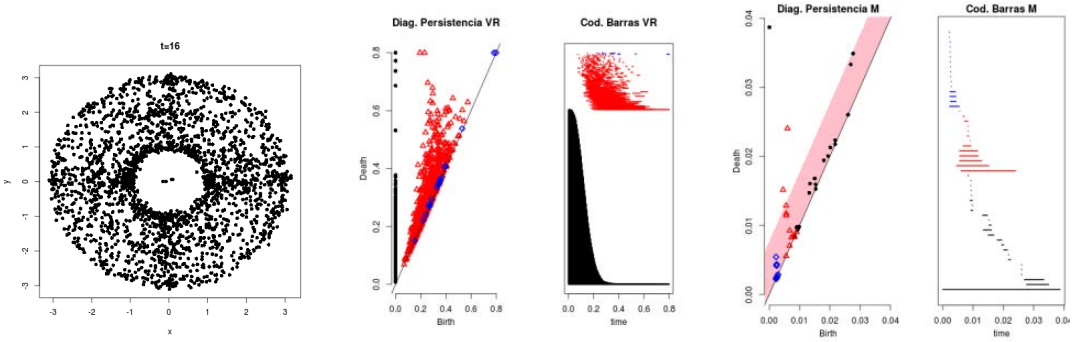
(a) 2500 datos sobre $\mathbb{T}^2$ (b) Filtración de Morse

Figura 2.33: Simulación sobre $\mathbb{T}^2$ con coeficientes de perturbación mediante procesos de Poisson de parámetro $\lambda = 0.2$ para $t = 4$.



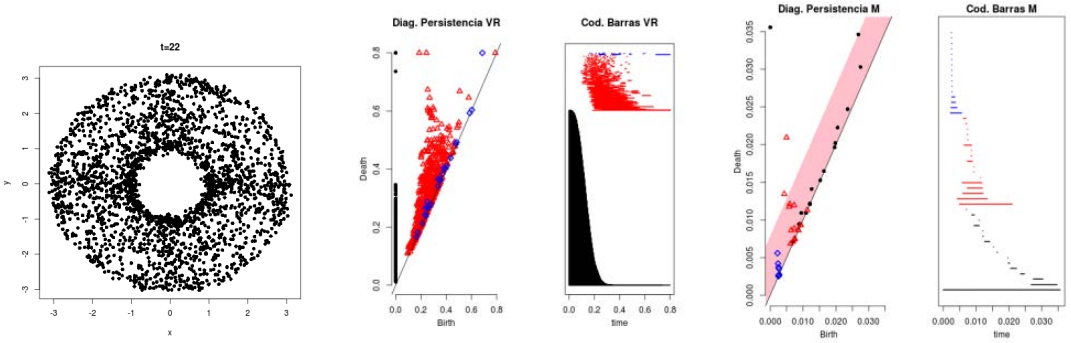
(a) 2500 datos sobre $\mathbb{T}^2$ (b) Filtración VR (c) Filtración de Morse

Figura 2.34: Simulación sobre $\mathbb{T}^2$ con coeficientes de perturbación mediante procesos de Poisson de parámetro $\lambda = 0.2$ para $t = 10$.



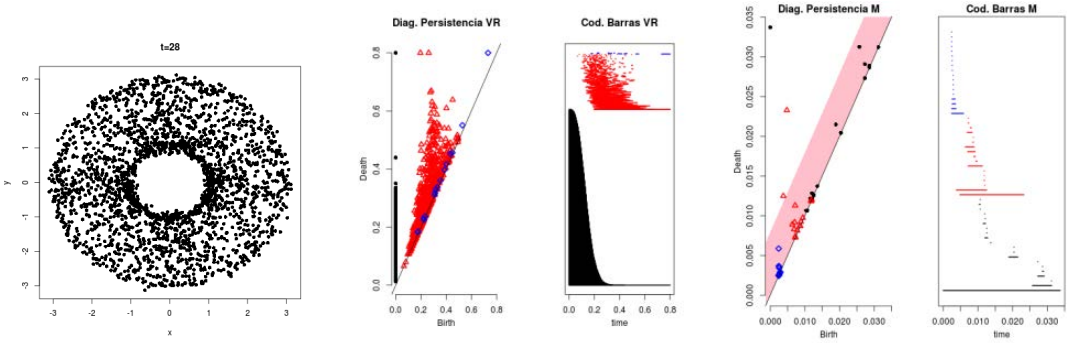
(a) 2500 datos sobre $\mathbb{T}^2$ (b) Filtración VR (c) Filtración de Morse

Figura 2.35: Simulación sobre $\mathbb{T}^2$ con coeficientes de perturbación mediante procesos de Poisson de parámetro $\lambda = 0.2$ para $t = 16$.



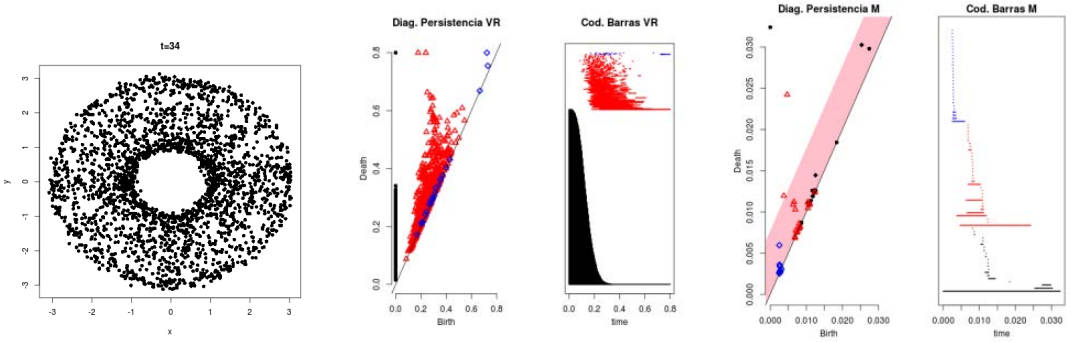
(a) 2500 datos sobre $\mathbb{T}^2$ (b) Filtración VR (c) Filtración de Morse

Figura 2.36: Simulación sobre $\mathbb{T}^2$ con coeficientes de perturbación mediante procesos de Poisson de parámetro $\lambda = 0.2$ para $t = 22$.



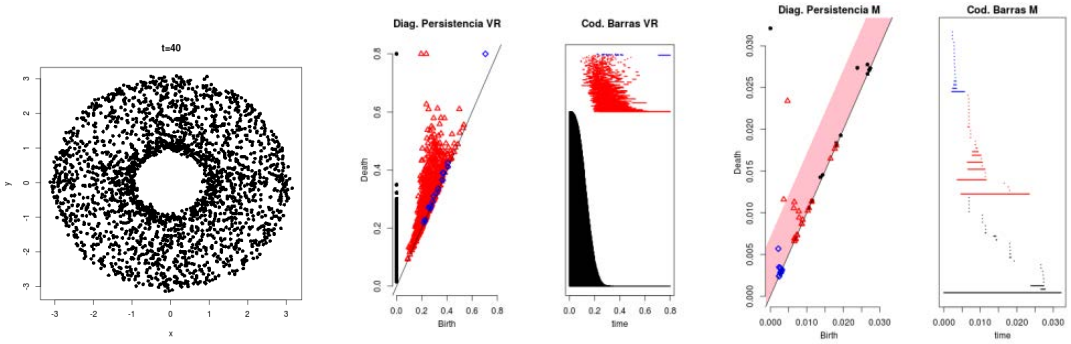
(a) 2500 datos sobre $\mathbb{T}^2$ (b) Filtración VR (c) Filtración de Morse

Figura 2.37: Simulación sobre $\mathbb{T}^2$ con coeficientes de perturbación mediante procesos de Poisson de parámetro $\lambda = 0.2$ para $t = 28$.



(a) 2500 datos sobre $\mathbb{T}^2$ (b) Filtración VR (c) Filtración de Morse

Figura 2.38: Simulación sobre $\mathbb{T}^2$ con coeficientes de perturbación mediante procesos de Poisson de parámetro $\lambda = 0.2$ para $t = 34$.



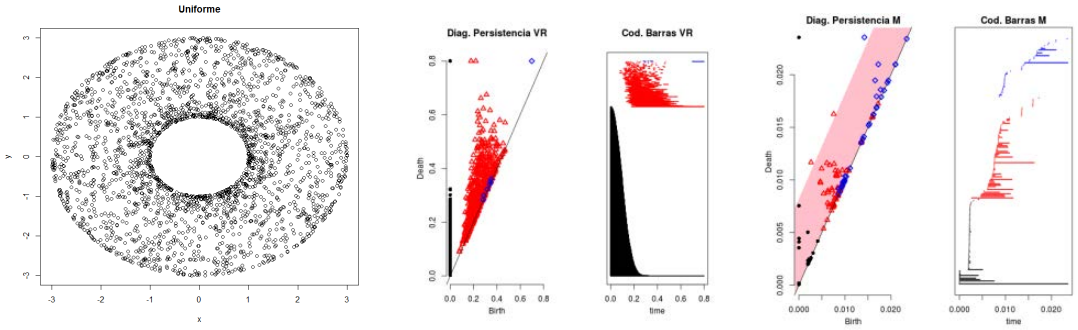
(a) 2500 datos sobre $\mathbb{T}^2$ (b) Filtración VR (c) Filtración de Morse

Figura 2.39: Simulación sobre $\mathbb{T}^2$ con coeficientes de perturbación mediante procesos de Poisson de parámetro $\lambda = 0.2$ para $t = 40$.

En las Figuras 2.27–2.34 se puede observar que para los tiempos el algoritmo MS no es capaz de detectar la concentración en ejes cartesianos como lo hacía en los casos del círculo $\mathbb{S}^1$ y la esfera $\mathbb{S}^2$. Se detecta una componente conexa en todos los tiempos mostrados. En el tiempo $t = 16$ se detectan los 2 agujeros $1-$dimensionales parte de la geometría del toro; a partir de este tiempo se mantienen los números de Betti $\beta_0 = 1$ y $\beta_1 = 2$. Sin embargo, el algoritmo es incapaz de encontrar el agujero $2-$dimensional para ningún tiempo. Probamos tiempos más grandes para ver si se encontraba el número de Betti $\beta_2 = 1$ sin éxito. Es complicado encontrar un valor óptimo para el ancho de banda del estimador kernel y al mismo tiempo un ancho de rejilla que ajuste de manera correcta sin usar completamente los recursos del servidor. Se encuentran características "similares" cuando se hace el cálculo de la homología usando los complejos testigo (ver Sección 3.2.2) con esta misma nube de datos en $\mathbb{T}^2$.

El algoritmo VR para el tiempo $t = 10$ detecta muchos agujeros $1-$dimensionales y un número alto de componentes conexas. Es a partir del tiempo $t = 16$ que se detecta los dos agujeros $1-$dimensionales ($\beta_1 = 2$). El número de Betti $\beta_2 = 1$ aparece de manera notable a partir del tiempo $t = 22$. Es importante mencionar que debemos regular con cuidado el valor máximo de la filtración, ya que un valor "grande" hace que el algoritmo agote los recursos de cómputo con los que contamos.

En la Figura 2.40 se muestra el cálculo de la homología para $\mathbb{T}^2$ mediante los algoritmos VR y MS. Se puede observar que en el caso de VR el comportamiento es muy parecido a lo que se tiene para el tiempos $t = 40$. En el caso de MS, se capturan de manera correcta los números de Betti $\beta_0 = 1,\ \beta_1 = 2,\ \beta_2 = 1$. Sin embargo, estas características están muy cercanas a la banda de confianza



(a) 2500 datos sobre $\mathbb{T}^2$ (b) Filtración VR (c) Filtración de Morse

Figura 2.40: Simulación sobre $\mathbb{T}^2$ con distribución uniforme.

En los Capítulos 3 y 4 se retomarán las nubes de datos usadas en estos ejemplos para ver el comportamiento de los otros algoritmos en cuanto a la detección de la concentración sobre los puntos cardinales.

3

Alternativas simpliciales a los complejos de Čech y Vietoris-Rips

El costo computacional para el caso de los complejos simpliciales de Čech y de Vietoris-Rips es del orden $O(n^3)$, donde n es el tamaño de la nube de puntos. Estos algoritmos "clásicos" llegan a calcular un sin número de subsimplejos (dependiendo de n) que incluso son de una dimensión más alta que el espacio ambiente de la muestra. En este Capítulo se presentan algunas alternativas simpliciales a las filtraciones de Čech y Vietoris-Rips. En la Sección 3.1 se presentan los antecedentes y la construcción de los complejos alfa, en donde el cálculo de la homología está regulado por un parámetro de "forma" α. En la Sección 3.2 se describe el marco teórico que ayuda a la construcción de los complejos testigo, así como una implementación de los mismos. En la Sección 3.2.2 se presentan ejemplos con los que se compara la efectividad de los complejos testigo con los algoritmos VR y MS, usando las mismas nubes de datos perturbadas que se presentan al final del Capítulo 2.

3.1. Complejos alfa

Un enfoque para evitar la problemática que tienen tanto los complejos de Čech como los de Vietoris-Rips respecto al crecimiento en el número de simplejos cuando el tamaño de muestra aumenta son los complejos alfa introducidos por Edelsbrunner y Mücke (1994). Los complejos alfa surgen a partir de una generalización de las $\alpha-$formas que presentaron Edelsbrunner, Kirkpatrick y Seidel (1983) mismas que nacen como una alternativa al cálculo de la envolvente convexa (presentada en el Capítulo 2) de una nube finita de puntos en $\mathbb{R}^2$. Son presentados por primera vez en el contexto de geometría computacional con la idea de obtener la "forma" de un conjunto finito de datos en el plano.

Para definir las α-formas es necesario definir primero el concepto de un disco de manera general. Se define un *disco generalizado de radio* $1/\alpha$ como:

- Disco de radio $1/\alpha$, si $\alpha > 0$.
- El complemento de un disco de radio $-1/\alpha$, si $\alpha < 0$.
- Un semiplano, si $\alpha = 0$.

Definición 35. Dado un conjunto finito de puntos $S \subset \mathbb{R}^2$ y para un valor α arbitrario, se define la $\alpha-$envolvente como la intesección de todos los discos generalizados de radio $1/\alpha$ que contienen a S.

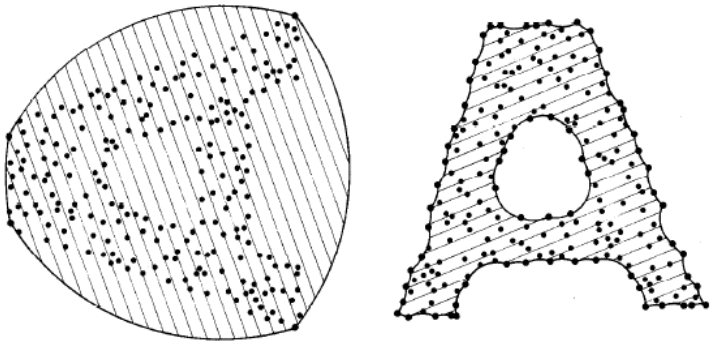


Figura 3.1: Representación de $\alpha-$evolvente positiva (izquierda) y negativa (derecha) (Edelsbrunner et al., 1983).

Para asegurar la existencia de un disco con estas características se recurre a Jung (1899), mismo que prueba que un valor para $1/\alpha$ no menor a $3^{-1/2}$ veces el diámetro de S es suficiente para cubrir a dicho conjunto. Se define también la evolvente α para el caso de valores negativos.

Observación La α_1 evolvente está contenida en la α_2-evolvente si $\alpha_1 \leqslant \alpha_2$.

Un concepto importante para definir las $\alpha-$formas es el de los puntos $\alpha-$extremos, decimos que un punto $p \in S \subset \mathbb{R}^2$ es $\alpha-$*extremo* si existe un disco generalizado cerrado de radio $1/\alpha$ tal que p está en la frontera del mismo y además contiene a S. Si dos puntos $\alpha-$extremos p y q están en la frontera del mismo disco generalizado decimos que son $\alpha-$vecinos.

Definición 36. Dado un conjunto finito de puntos $S \subset \mathbb{R}^2$ y un número arbitrario α, la $\alpha-$forma de S es el grafo lineal cuyos vértices son los puntos $\alpha-$extremos y las aristas son las que unen a los $\alpha-$vecinos respectivos.

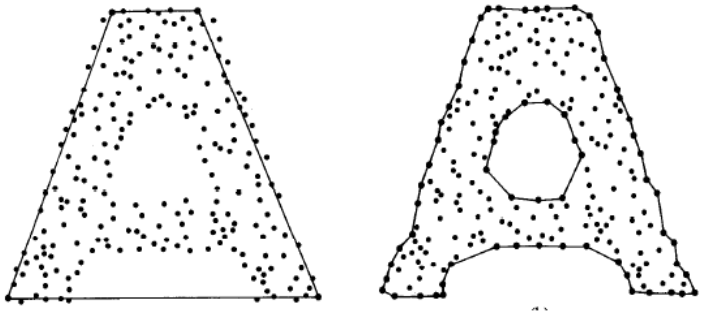


Figura 3.2: Representación de $\alpha-$formas positiva (izquierda) y negativa (derecha) (Edelsbrunner et al., 1983).

Con esta idea en mente se pueden definir los complejos alfa, pero primero es necesario definir unos conceptos adicionales.

Definición 37. Sea P un conjunto discreto de puntos en $\mathbb{R}^d$. La *triangulación de Delaunay* $\mathcal{D}(P)$ es un complejo simplicial $\mathcal{D}(P) \subseteq 2^P$ que satisface:

- Dado cualquier punto $p \in P$ y cualquier simplejo $\sigma \in \mathcal{D}(P)$ con $\dim(\sigma) = d$, p no está contenido en la hiperesfera circunscrita de los $d+1$ vértices de σ.
- Si C es la envolvente convexa de P, entonces

$$\bigcup_{\sigma \in \mathcal{D}(P):\dim(\sigma)=n} \sigma = C.$$

Definición 38. Sea P un subconjunto finito de $\mathbb{R}^d$. Para un punto $p \in P$, la región de Voronoi R_p es el conjunto de puntos en $\mathbb{R}^d$ que están más cercanos a p que a cualquier otro punto $q \in P$. Esto es:

$$R_p = \{r \in \mathbb{R}^d | \forall q \in P \backslash p, d(p,r) \leqslant d(q,r)\}.$$

Definimos entonces el diagrama de Voronoi $\mathcal{V}(P)$ como la colección de las regiones de Voronoi

$$\mathcal{V}(P) = \{R_p | p \in P\}.$$

El diagrama de Voronoi es el nervio de la triangulación de Delaunay, de donde podemos dar lugar al complejo de Delaunay de manera análoga en que el nervio de una cubierta euclideana da paso al complejo de Čech.

El complejo alfa es similar al de Čech en el sentido de que el método de inclusión de los simplejos es similar, excepto que los simplejos disponibles para la inclusión están restringidos a la triangulación de Delaunay conformada por la nube de puntos como vértices.

Supongamos $P \subseteq \mathbb{R}^d$ y sea $\mathcal{D}(P)$ y $\mathcal{V}(P)$ la triangulación de Delaunay y el diagrama de Voronoi, respectivamente. El complejo alfa de $\mathcal{A}(P, \alpha)$ es obtenido al tomar cada región de Voronoi R_p e intersectarla con $B(p, \alpha)$, la bola de radio α centrada en p. Un simplejo

$\alpha = [p_1, p_2, ..., p_k]$ está contenido en $\mathcal{A}(P, \alpha)$ si todas las células resultantes asociadas a $p_1, p_2, ..., p_k$ tienen intersecciones no vacías a pares. Denotamos el diagrama alfa como

$$Q(P, \alpha) = \bigcup_{i=1}^{|P|} B(p_i, \alpha) \cap R_p,$$

el cual es el dual del complejo alfa $\mathcal{A}(X, \alpha)$.

Nota: Un espacio dual $V*$ de un espacio vectorial V, es el conjunto de todas las funciones lineales de V a su respectivo campo de coeficientes F. Es decir $V = \{\psi : \ \psi : V \to F\}_\psi$ es lineal·

Figura 3.3: El diagrama de Voronoi (izquierda) es el dual de la triangulación de Delaunay (centro). El complejo alfa $A(P, \alpha)$ (derecha) se obtiene al intersectar cada región de Voronoi R_p con la bola $B(p, \alpha)$ (Hennigan, 2015).

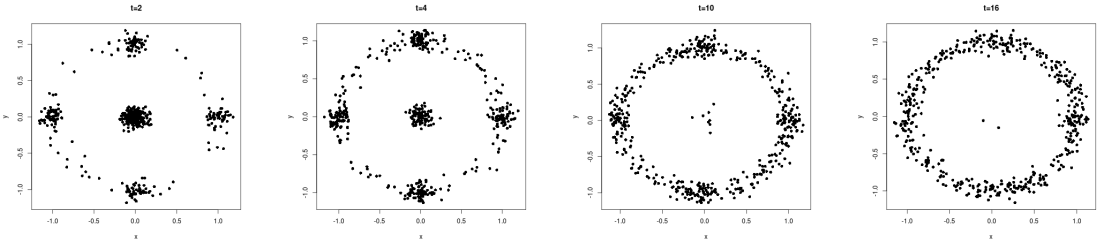
Una buena característica del complejo alfa es que tiene el mismo tipo de homotopía que el complejo de Čech, pero se puede calcular con mucho menos simplejos y el número de simplejos en el complejo no se nos sale de las manos al incrementar el parámetro de la filtración, dado que el complejo maximal es simplemente la triangulación de Delaunay.

Para poder utilizar el complejo alfa, es necesario calcular primero la triangulación de Delaunay de un conjunto de puntos que está relacionado de manera muy cercana a calcular la envolvente convexa de un conjunto de vértices. Pero existe un problema al calcular la triangulación de Delaunay para un conjunto de m puntos $P \subset \mathbb{R}^d$, ya que es equivalente al problema de encontrar la envolvente convexa para m puntos $P' \subset \mathbb{R}^d$, donde P' se obtiene al proyectar todos los puntos $p \in P$ sobre un paraboloide en $\mathbb{R}^{d+1}$.

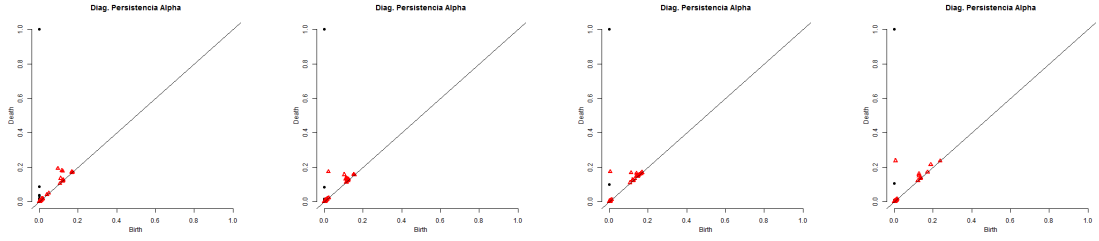
De todos los algoritmos disponibles, la mayoría tienden a tener complejidad de tiempo exponencial con respecto a la dimensión de los datos. Esto hace que el complejo alfa sea de igual manera intratable en conjuntos de datos altamente dimensionales.

3.1.1. Ejemplos

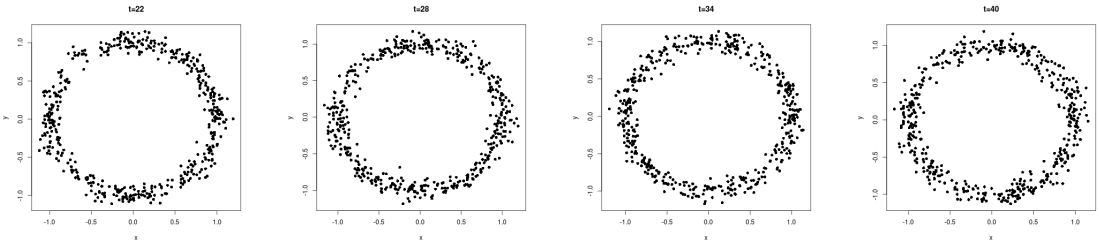
En los ejemplos presentados a continuación estamos utilizando las nubes de datos de los ejemplos de la Sección 2.6. En este caso solamente mostramos diagramas de persistencia, pues el algoritmo ha sido recientemente implementado en la paquetería TDA de R y aún no se incluyen los códigos de barra.



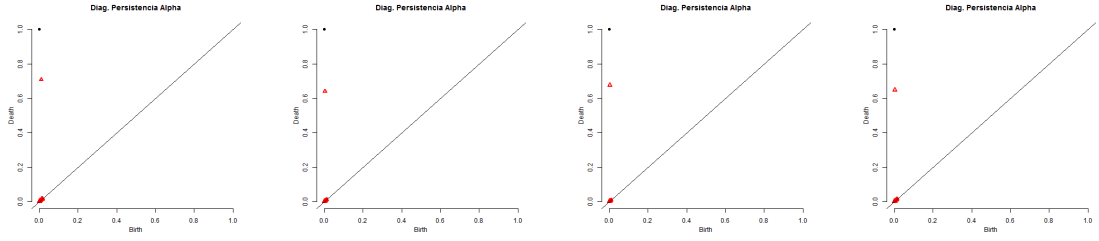
(a) $t = 2$ (b) $t = 4$ (c) $t = 10$ (d) $t = 16$



(e) $t = 2$ (f) $t = 4$ (g) $t = 10$ (h) $t = 16$



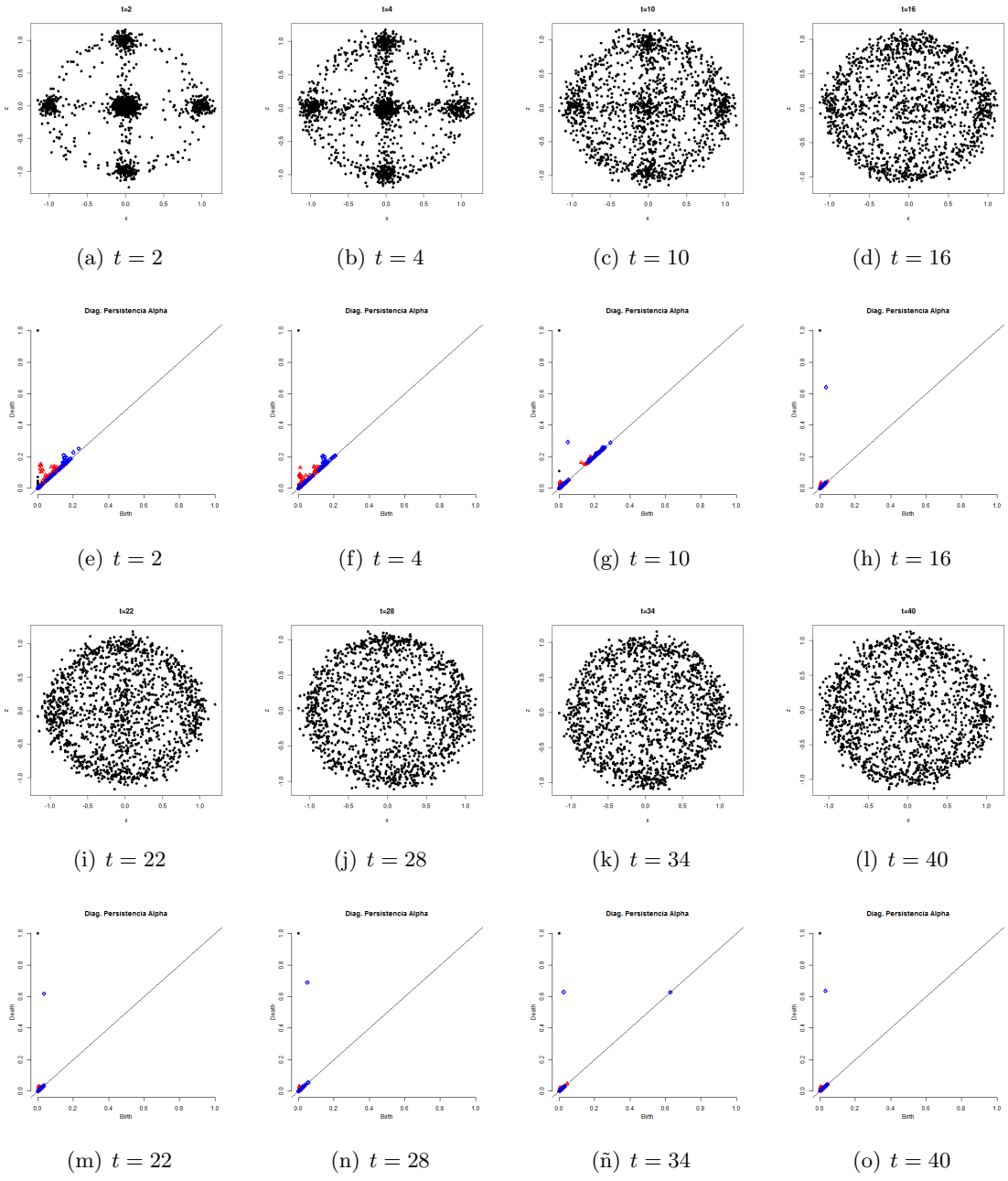
(i) $t = 22$ (j) $t = 28$ (k) $t = 34$ (l) $t = 40$



(m) $t = 22$ (n) $t = 28$ (ñ) $t = 34$ (o) $t = 40$

Figura 3.4: 500 datos sobre $\mathbb{S}^1$ con coeficientes de perturbación mediante procesos de Poisson de parámetro $\lambda = 0.2$.

Podemos observar en la Figura 3.4 que el algoritmo de los complejos alfa encuentra una componente conexa desde el tiempo $t = 2$ hasta el último tiempo mostrado. No es capaz de detectar los puntos que están al centro de manera notable (outliers). El ciclo 1-dimensional aparece cuando los puntos al centro del círculo ya no están, esto es a partir del tiempo $t = 22$ hasta el último tiempo mostrado. No detecta concentraciones de puntos en ningún caso.



(a) $t = 2$ (b) $t = 4$ (c) $t = 10$ (d) $t = 16$

(e) $t = 2$ (f) $t = 4$ (g) $t = 10$ (h) $t = 16$

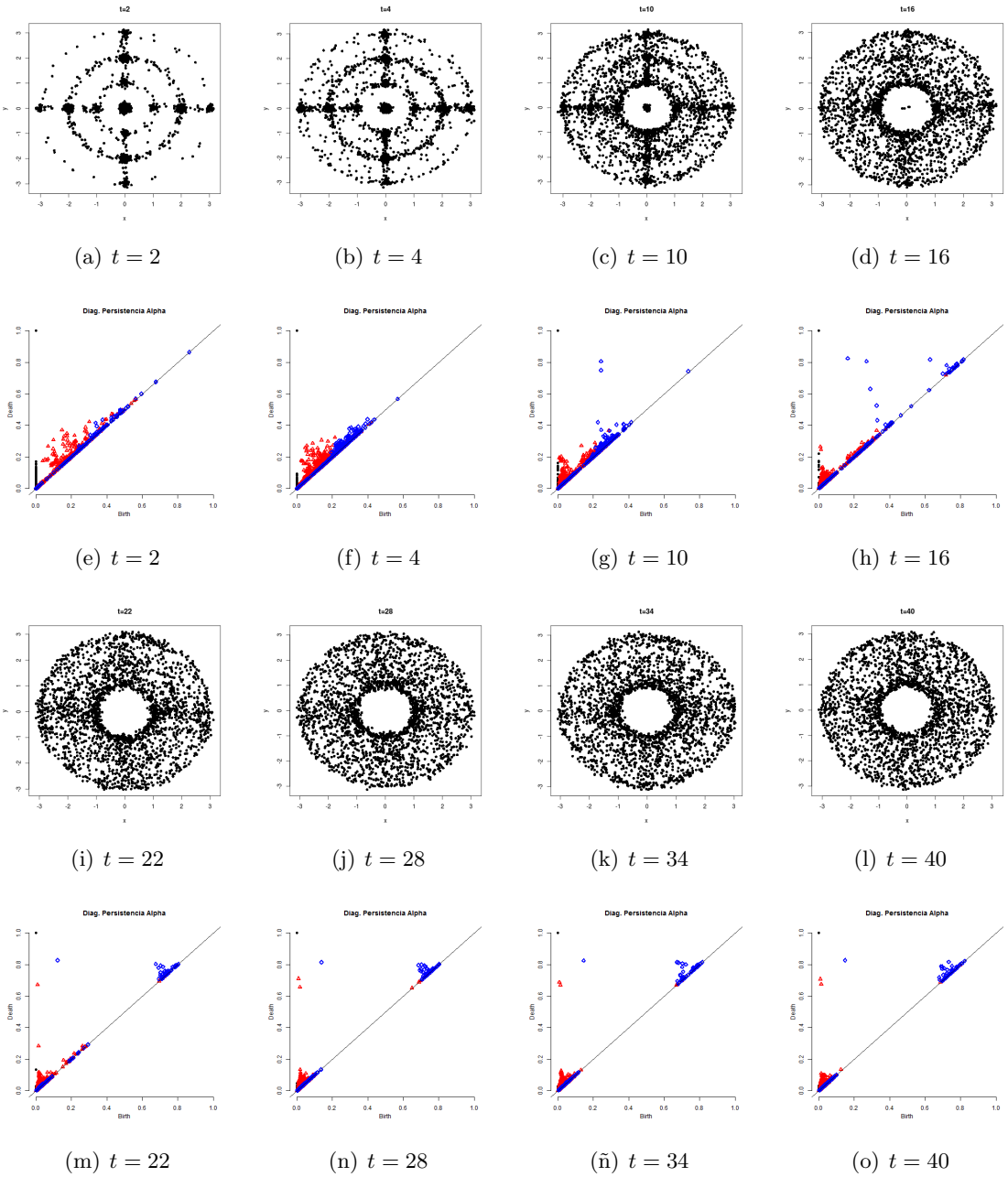
(i) $t = 22$ (j) $t = 28$ (k) $t = 34$ (l) $t = 40$

(m) $t = 22$ (n) $t = 28$ (ñ) $t = 34$ (o) $t = 40$

Figura 3.5: 1500 datos sobre $\mathbb{S}^2$ con coeficientes de perturbación mediante procesos de Poisson de parámetro $\lambda = 0.2$.

Al igual que en $\mathbb{S}^1$, en la Figura 3.10 podemos observar cómo el algoritmo de los complejos alfa detecta una componente conexa desde el tiempo $t = 2$ hasta el último tiempo mostrado. En este caso, el ciclo $2-$dimensional empieza a destacar en el diagrama desde el tiempo $t = 10$, siendo en el tiempo $t = 16$ donde podemos decir que se encuentra el número de Betti $\beta_2 = 1$. El algoritmo no detecta la concentración en los distintos puntos de la esfera, ni la concentración alrededor de cero. Tampoco detecta los $1-$ciclos que se

forman en los cuadrantes de la esfera en el tiempo $t = 2$ (y posiblemente en el tiempo $t = 4$).



(a) $t = 2$ (b) $t = 4$ (c) $t = 10$ (d) $t = 16$

(e) $t = 2$ (f) $t = 4$ (g) $t = 10$ (h) $t = 16$

(i) $t = 22$ (j) $t = 28$ (k) $t = 34$ (l) $t = 40$

(m) $t = 22$ (n) $t = 28$ (ñ) $t = 34$ (o) $t = 40$

Figura 3.6: 2500 datos sobre $\mathbb{T}^2$ con coeficientes de perturbación mediante procesos de Poisson de parámetro $\lambda = 0.2$.

De manera similar a los dos casos anteriores, en la Figura 3.6 se puede observar que el algoritmo detecta una componente conexa desde $t = 2$ hasta $t = 40$, no detecta las concentraciones en los ejes cartesianos ni alrededor de 0. El algoritmo detecta ruido $1-$ dimensional desde el tiempo $t = 2$ y es hasta el tiempo $t = 28$ que podemos decir que

el algoritmo encuentra los dos 1−ciclos "característicos" del toro. Además, se puede ver cómo en los tiempos $t = 10,\ 16$ se detectan al menos dos ciclos 2−dimensionales (podría deberse al círculo que se encuentra dentro del toro), y es hasta el tiempo $t = 22$ que se encuentra el 1−ciclo que forma parte de la geometría de $\mathbb{T}^2$.

3.2. Complejos testigo

Al igual que los complejos alfa, los complejos testigo están motivados por los complejos de Delaunay, así como en una submuestra de la nube completa de puntos llamada *puntos de referencia*. La idea de este tipo de complejos es reducir el costo computacional generando complejos simpliciales a partir de los puntos de referencia y tomando como testigos de la existencia de simplejos el restante en la nube de puntos. La implementación actual de los complejos testigo nos permite obtener una familia anidada de éstos, lo cual nos ayuda a calcular la homología persistente.

Podemos considerar a los complejos testigo como una aproximación a la triangulación de Delaunay restringida, la ventaja es que su construcción evita la "maldición" de la dimensionalidad a la que se enfrentan los cálculos de Delaunay. Una de las ventajas principales de los complejos testigo es que están definidas para cualquier espacio métrico, no necesariamente euclideano.

La selección de los *puntos de referencia* $\mathcal{L}$ se hace utilizando la selección MaxMin que describimos en la Sección 3.2.1. Se define el complejo testigo (fuerte) como sigue:

Definición 39. Sea $\mathcal{L} = \{\ell_1, \ell_2, ..., \ell_k\}$, D la matriz de distancias de tamaño $k \times n$ entre el conjunto con los k puntos de referencia seleccionados y los n puntos de una nube de puntos P proveniente de un espacio topológico X. Generamos el complejo $W_\infty(D)$ como sigue:

- La arista $\sigma = [ab]$ pertenece a $W_\infty(D)$ si y sólo si existe un punto ℓ_i para $1 \leqslant i \leqslant N$ tal que $d(a, \ell_i)$ y $d(b, \ell_i)$ son las dos entradas más pequeñas en la $i-$ésima columna de D.

- Mediante inducción en p: supóngase que todas las caras del $p-$simplejo σ definido como $\sigma = [a_0 a_1 \cdots a_p]$ pertenecen a $W_\infty(D)$. Entonces, σ pertenece a $W_\infty(D)$ si y sólo si existe un punto ℓ_i para $1 \leqslant i \leqslant N$ tal que $d(a_0, \ell_i), d(a_1, \ell_i), ..., d(a_p, \ell_i)$ son las $p + 1$ entradas más pequeñas de la $i-$ésima columna.

En cada caso ℓ_i es considerado un "testigo" de la existencia de σ.

En la siguiente figura mostramos un ejemplo de un complejo testigo fuerte mediante la selección de 3 puntos de referencia para el caso de puntos tomados de un triángulo rectángulo.

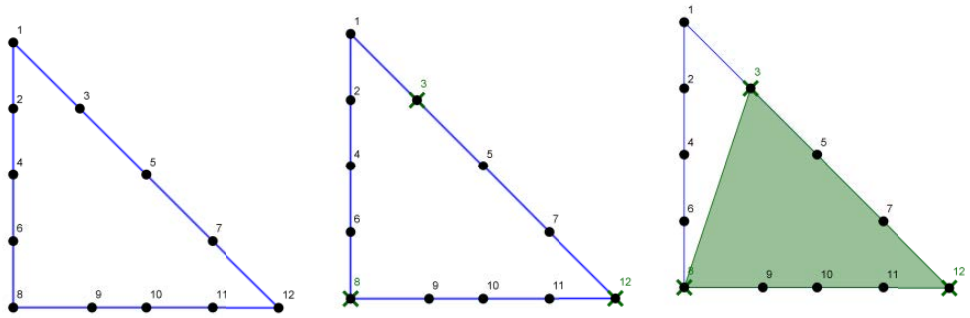


Figura 3.7: Ejemplo de la construcción de un complejo testigo (verde) con 3 puntos de referencia.

Así como el complejo Vietoris-Rips es una relajación en la definición del complejo de Čech, podemos relajar de manera análoga la definición anterior. Lo denotamos mediante $W_1(D)$ y se define como sigue:

Definición 40. (a) $W_1(D)$ tiene el mismo $1-$esqueleto que $W_\infty(D)$.

(b) El $p-$simplejo $\sigma = [a_0 a_1 ... a_p]$ pertenece a $W_1(D)$ si y sólo si todos sus vértices pertenecen a $W_1(D)$

$W_1(D)$ es el complejo simplicial más grande que tiene los mismos vértices y aristas que $W_\infty(D)$. Resulta que $W_\infty(D)$ es demasiado exigente computacionalmente en la práctica, por lo que se opta por por trabajar con $W_1(D)$ que en adelante se denotará como $W(D)$.

Para calcular la persistencia en los complejos testigo es necesario definir una familia de éstos que satisfaga el requisito de multiresolución con respecto al parámetro que llamamos tiempo ε. Así pues, supóngase que D es una matriz $k \times n$ de distancias, como se definió antes. Para cada entero no negativo ν se construye una familia de complejos simpliciales $W(D; \varepsilon, \nu)$ donde $\varepsilon \in [0, \infty]$. El conjunto de vértices de $W(D; \varepsilon, \nu)$ es $\{\ell_1, \ell_2, ..., \ell_k\}$.

Definición 41.

- Si $\nu = 0$, entonces para $i = 1, 2, ...n$ se define $m_i = 0$.
- Si $\nu > 0$, entonces para $i = 1, 2, ..., n$ se define m_i como la $\nu-$ésima entrada más pequeña de la $i-$ésima columna de D.
- La arista $\sigma = [ab]$ pertenece a $W(D; \varepsilon, \nu)$ si y sólo si existe un testigo ℓ_i para $i \in \{1, 2, ..., n\}$ tal que

$$\text{máx}(D(a, \ell_i), D(b, \ell_i)) \leqslant \varepsilon + m_i.$$

- El $p-$simplejo $\sigma = [a_0 a_1 \cdots a_p]$ pertenece a $W(D; \varepsilon, \nu)$ si y sólo si todas sus caras pertenecen a $W(D; \varepsilon, \nu)$; equivalentemente si y sólo si existe un testigo ℓ_i para $1 \leqslant i \leqslant n$ tal que

$$\text{máx}(D(a_0, \ell_i), D(a_1, \ell_i), ..., D(a_p, \ell_i)) \leqslant \varepsilon + m_i$$

Se satisface la identidad $W(D; 0, 2) = W(D) = W_1(D)$. De Silva y Carlsson (2004) mencionan los casos en que $\nu = 0, 1, 2$ como de particular importancia pues se satisfacen las siguientes propiedades.

- $\nu = 0$: La familia de complejos $W(D; \varepsilon, 0)$ está cercanamente relacionada a la familia de complejos Rips $R(L; \varepsilon)$. Específicamente, se cumplen las siguientes inclusiones:

$$W(D; \varepsilon, 0) \subseteq \text{Rips}(L; 2\varepsilon) \subseteq W(D; 2\varepsilon, 0).$$

- $\nu = 1$: Se puede interpretar como proveniente de una familia de cubiertas del espacio X mediante regiones de Voronoi que rodean cada punto de referencia, las cuales se traslapan cuando $\varepsilon \to \infty$.

- $\nu = 2$: Recuérdese que se tenía la siguiente identidad en $\varepsilon = 0$

$$W(D; 0, 2) = W(D).$$

Al llevar a cabo el cálculo de la persistencia con $\nu = 2$, se puede observar que esta familia de complejos da intervalos de persistencia más claros, con poco "ruido" en los códigos de barra. Una explicación de esto se debe a la identidad recién mencionada, pues el complejo simplicial es esencialmente el mismo que se menciona en la Definición 39 cuando $\varepsilon = 0$, por lo que es suficente contar con incrementos pequeños en el valor de ε.

3.2.1. Selección MaxMin

Una forma alternativa de elegir una muestra es basándose en la mayor distancia posible que hay de un primer punto seleccionado a otro y elegir así una muestra representativa mediante un proceso iterativo, este es el proceso *MaxMin* que se define como sigue:

- Selecciónese $\ell_1 \in X$ de manera aleatoria.

- Inductivamente, si $\ell_1, \ell_2, ..., \ell_{i-1}$ han sido elegidos, sea $\ell_i \in X \backslash \{\ell_0, \ell_1, ..., \ell_{i-1}\}$ el punto que maximiza la función

$$x \mapsto \text{mín}\{D(x, \ell_1), D(x, \ell_2), ..., D(x, \ell_{i-1}\},$$

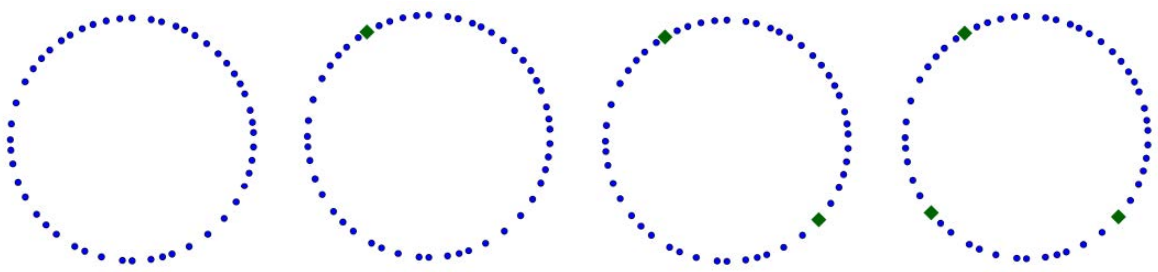
  donde D es la métrica.

- Continúese hasta que se hayan elegido el número deseado de puntos.

Los puntos elegidos mediante *MaxMin* tienden a estar más espaciados, pero son susceptibles a tomar *outliers*. El número de puntos de referencia a elegir deben ser tales que la razón n/k esté acotada de manera superior. De Silva y Carlsson (2004) sugieren esta cota como 20 de manera heurística debido a los experimentos realizados en su artículo.

Nota: Un *outlier* es un dato que cae fuera del comportamiento "normal" del resto de la nube de puntos.

En la siguiente figura ilustramos con detalle la idea principal del método MaxMin:



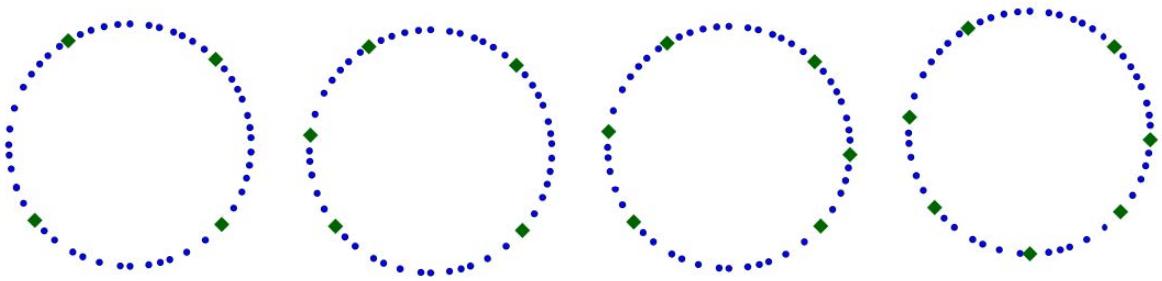
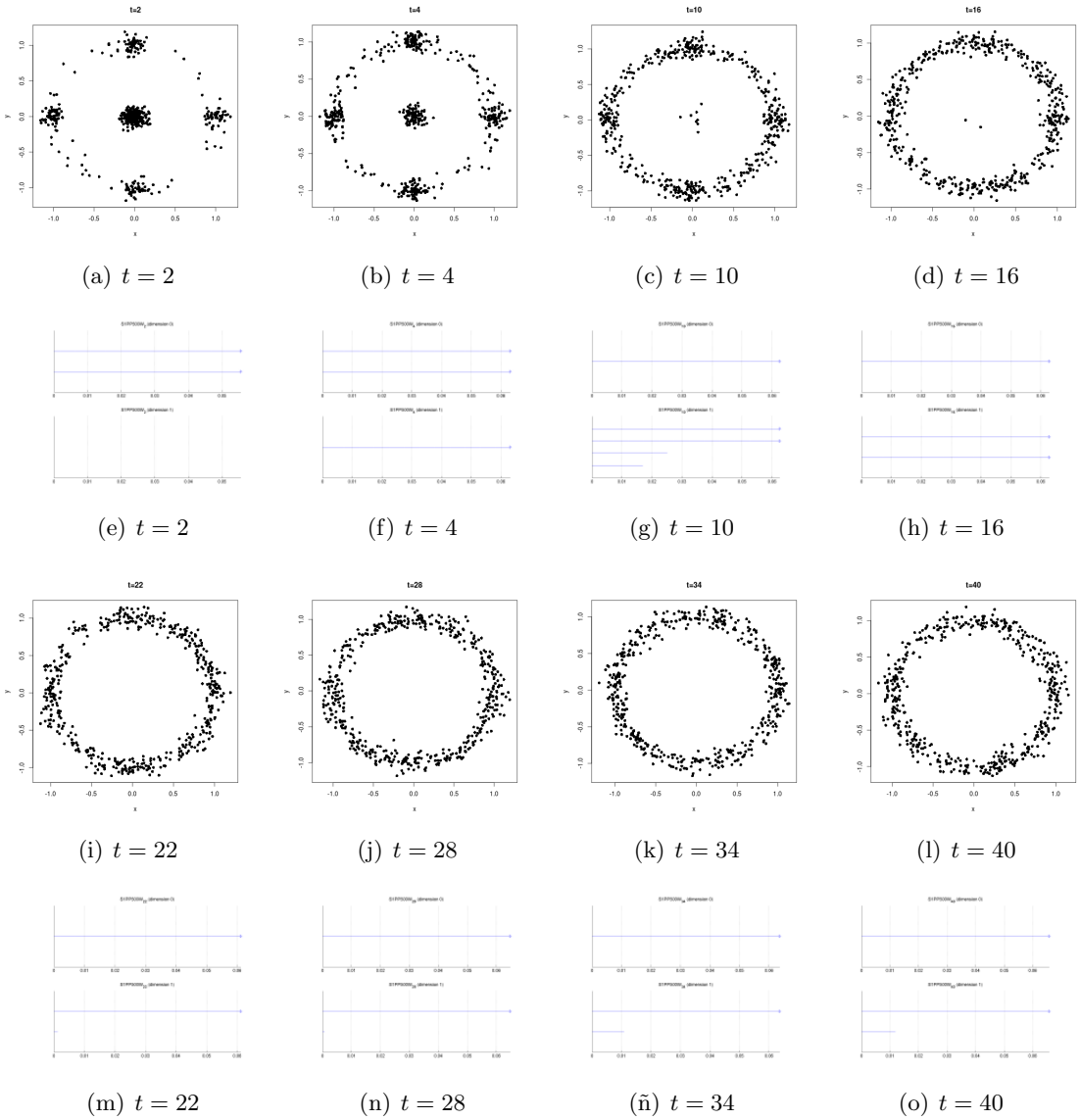


Figura 3.8: Ejemplo de selección de 8 puntos mediante MaxMin de una muestra de $\mathbb{S}^1$.

Es posible elegir los puntos de referencia de manera aleatoria, con la "desventaja" de que estos no se encontrarían equi-espaciados como sucede cuando se utiliza la selección MaxMin.

3.2.2. Ejemplos

En los ejemplos que presentamos a continuación retomamos las nubes de datos de los ejemplos de la Sección 2.6. Se presenta un código de barras para mostrar la persistencia, el algoritmo devuelve como resultado únicamente este tipo de resumen pues se incluye muy poco ruido. La lectura de los códigos de barra en los complejos testigo es más fácil a comparación de los obtenidos mediante el algoritmo VR. La cantidad k de puntos de referencia se toman siguiendo la razón $n/k = 20$, como sugieren De Silva y Carlsson (2004). Para el radio de la filtración se toma como parámetro la distancia máxima que existe entre el conjunto de los puntos de referencia y el resto de la nube de datos.



(a) $t = 2$ (b) $t = 4$ (c) $t = 10$ (d) $t = 16$

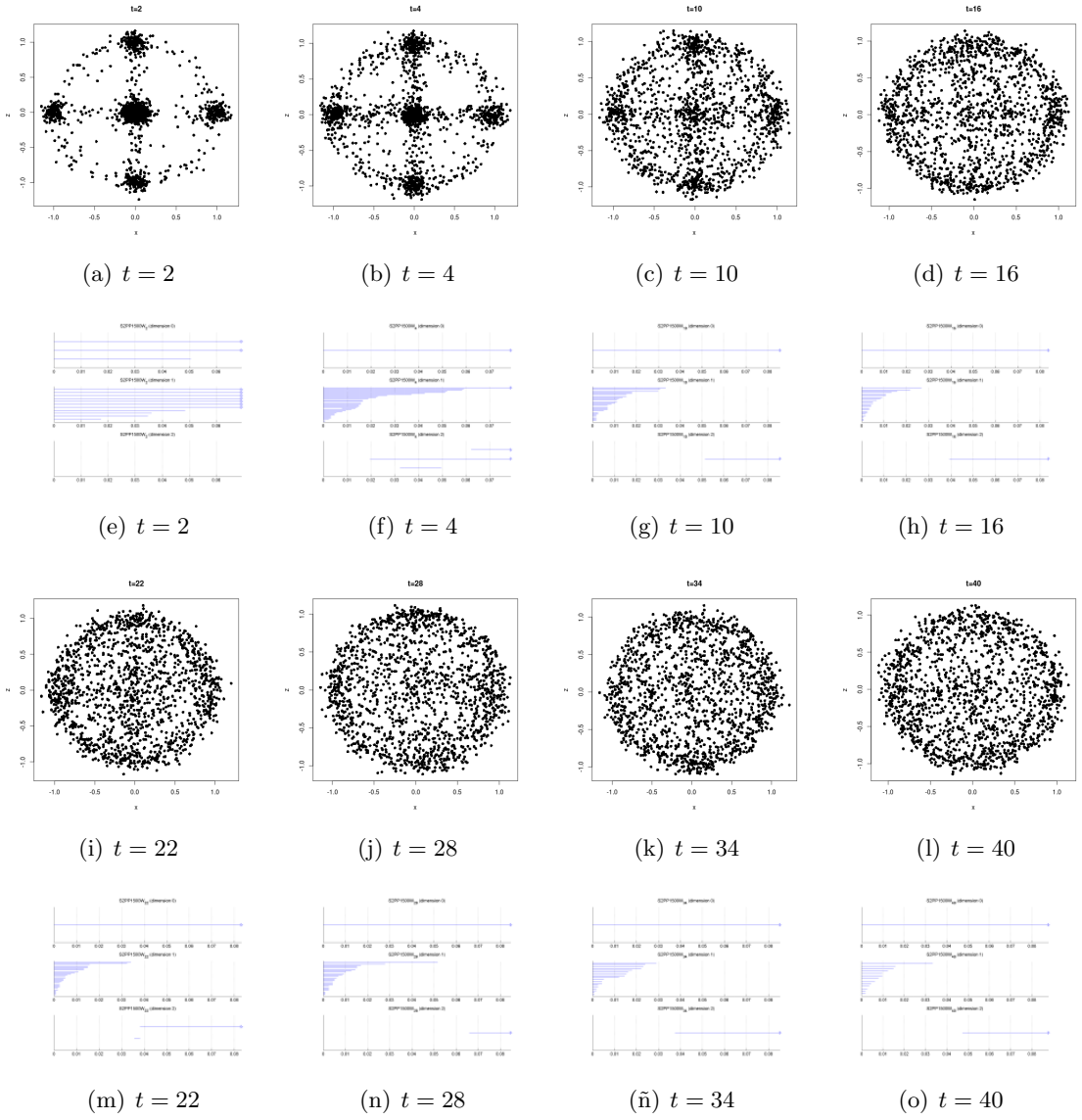
(e) $t = 2$ (f) $t = 4$ (g) $t = 10$ (h) $t = 16$

(i) $t = 22$ (j) $t = 28$ (k) $t = 34$ (l) $t = 40$

(m) $t = 22$ (n) $t = 28$ (ñ) $t = 34$ (o) $t = 40$

Figura 3.9: 500 datos sobre $\mathbb{S}^1$ con coeficientes de perturbación mediante procesos de Poisson de parámetro $\lambda = 0.2$.

En la Figura 3.9 se puede observar que el algoritmo de los complejos testigo detecta dos componentes conexas (parte del círculo y los puntos centrales) para los tiempos $t = 2,\ 4$, pero el ciclo lo detecta justo en $t = 4$. Para los tiempos $t = 10,\ 16$ sucede algo interesante, los puntos centrales "engañan" al algoritmo pues detecta la existencia de dos círculos de los cuales sólo existe uno. A partir del tiempo $t = 22$ se encuentran de manera correcta los números de Betti $\beta_0 = 1,\ \beta_1 = 1$.



(a) $t = 2$ (b) $t = 4$ (c) $t = 10$ (d) $t = 16$

(e) $t = 2$ (f) $t = 4$ (g) $t = 10$ (h) $t = 16$

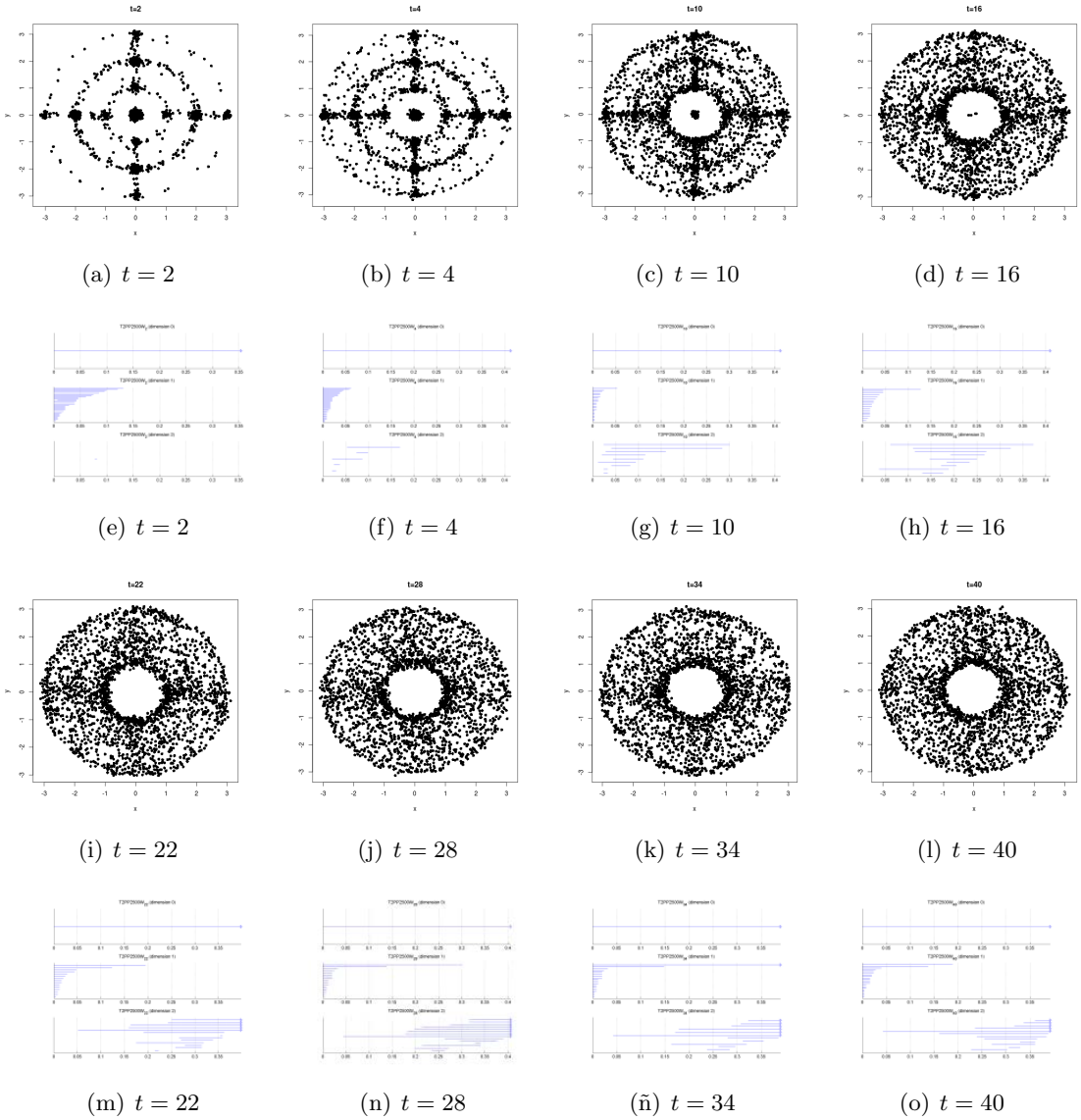
(i) $t = 22$ (j) $t = 28$ (k) $t = 34$ (l) $t = 40$

(m) $t = 22$ (n) $t = 28$ (ñ) $t = 34$ (o) $t = 40$

Figura 3.10: 1500 datos sobre $\mathbb{S}^2$ con coeficientes de perturbación mediante procesos de Poisson de parámetro $\lambda = 0.2$.

En la Figura 3.10 podemos observar cómo el algoritmo detecta 3 componentes conexas, pero no logra captar la concentración en ejes cartesianos. Para los tiempos $t = 4,\ 10$ se encuentra la componente conexa que de la esfera $\mathbb{S}^2$, misma que se mantiene hasta $t = 40$.

Sin embargo, para los tiempos $t = 2,\ 4$ se detectan varios agujeros $1-$dimensionales, lo cual puede deberse a los espacios existentes entre los cuadrantes de la esfera. Este ruido $1-$dimensional se mantiene a lo largo de todos los tiempos mostrados en la Figura 3.10. A partir de $t = 4$ aparece un hueco $2-$dimensional con algo de ruido topológico, pero a partir de este tiempo $\beta_2 = 1$ se mantiene hasta el último tiempo mostrado.



(a) $t = 2$ (b) $t = 4$ (c) $t = 10$ (d) $t = 16$

(e) $t = 2$ (f) $t = 4$ (g) $t = 10$ (h) $t = 16$

(i) $t = 22$ (j) $t = 28$ (k) $t = 34$ (l) $t = 40$

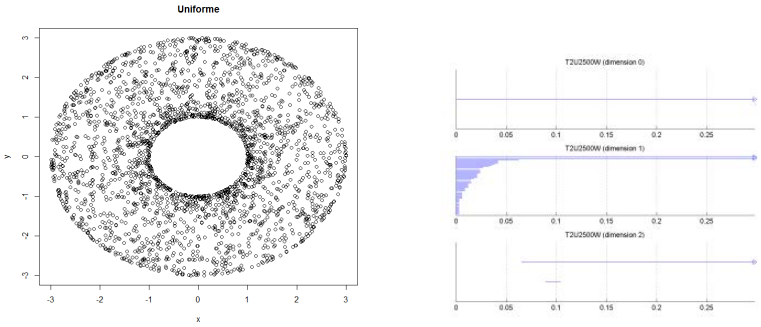
(m) $t = 22$ (n) $t = 28$ (ñ) $t = 34$ (o) $t = 40$

Figura 3.11: 2500 datos sobre $\mathbb{T}^2$ con coeficientes de perturbación mediante procesos de Poisson de parámetro $\lambda = 0.2$.

En la Figura 3.11 se puede observar que desde el tiempo $t = 2$ el algoritmo detecta una sola componente conexa y ruido $1-$dimensional. Es hasta el tiempo $t = 22$ cuando aparecen de manera notable los dos agujeros $1-$dimensionales que forman parte del toro. Esto se mantiene hasta $t = 40$. En el caso del agujero $2-$dimensional, el algoritmo

presenta problemas en encontrarlos de manera correcta. Incluso el número de agujeros en esta dimensión empieza a aumentar desde el tiempo $t = 10$. La problemática con el toro puede deberse a diversos factores como lo son, el ancho de banda h del estimador kernel, la resolución de la rejilla donde se evalúa el estimador o combinaciones de éstos. Se intentaron combinaciones óptimas mediante un cálculo de h usando un algoritmo de validación cruzada (ucv), pero el efecto que tuvo fue hacer uso total de los recursos del servidor.

En la Figura 3.12 mostramos el cálculo de la homología persistente en $\mathbb{T}^2$ por medio del algoritmo de los complejos testigo. En este caso los resultados obtenidos en el código de barras son los que esperaríamos en los tiempos más grandes. Cabe señalar que cuando la distribución es uniforme, el algoritmo captura de manera correcta los números de Betti del toro $\mathbb{T}^2$ ($\beta_0 = 1,\ \beta_1 = 2,\ \beta_2 = 1$).



(a) 2500 datos sobre $\mathbb{T}^2$ (b) Filtración VR

Figura 3.12: Simulación sobre $\mathbb{T}^2$ con distribución uniforme.

Se puede ver en estos ejemplos que al contrario del VR, el algoritmo de los complejos testigo no tiene ningún problema al correr sobre tiempos pequeños en la aproximación a la distribución uniforme en las tres variedades $\mathbb{S}^1$, $\mathbb{S}^2$ y $\mathbb{T}^2$. Sin embargo, es necesario "limpiar" los datos de *outliers* como lo sugieren Bubenik et al. (2010) para evitar "engañar" al algoritmo como sucedió con las perturbaciones hechas en la distribución uniforme de estas variedades.

La sugerencia es tomar una vecindad del mismo tamaño para cada punto en la nube de datos, a partir de esto hacer un conteo de puntos dentro de la misma y si el número no pasa cierto umbral, el punto se elimina. Se intentó implementar esta sugerencia en el caso donde hay fuertes concentraciones en algunas regiones de $\mathbb{S}^2$, sin embargo implementar ese método nos eliminó demasiados datos de modo que la nube resultante no era muy informativa. Esta situación nos lleva a calibrar lo "grueso" de la vecindad así como el umbral mínimo de puntos para no perder información valiosa.

4
Mapper

La propuesta de Mapper nació como una alternativa adicional a los complejos descritos en los Capítulos 2 y 3 pues como se ha mencionado varias veces, su costo computacional puede llegar a ser alto. El algoritmo Mapper se basa en elementos tales como el agrupamiento y algunas funciones que describen una nube de datos dada. Estos grupos son vértices de un complejo simplicial y se crean simplejos de dimensión mayor dependiendo de cuántos datos comparten entre ellos. Esto reduce de manera sorprendente el tiempo de cálculo y nos da una buena representación de las características topológicas de esa nube de puntos.

Este capítulo está destinado a describir el sustento teórico del algoritmo, así como algunos ejemplos donde podemos describir las estructuras de algunas nubes de datos simuladas. En la Seccción 4.1 se introduce el concepto de agrupamiento jerárquico y tres métodos distintos que ilustran esta técnica. En la Seccción 4.2 se presenta la idea topológica que da lugar al algoritmo Mapper así como su implementación estadística usando la técnica de agrupamiento jerárquico. En la Sección 4.3 se incluyen dos implementaciones en Python de este algoritmo, las cuales están disponibles en la red, así como algunos ejemplos para ilustrar el uso del software.

En la Sección 4.4 ilustramos el uso de Mapper con las nubes de datos simuladas en el Capítulo 2, comparando los resultados obtenidos con los complejo,s VR, MS y testigo.

Finalmente, en la Sección 4.5 incluímos otro algoritmo que hace uso de técnicas de agrupamiento y componentes principales para la construcción de un complejo mediante árboles.

4.1. Agrupamiento jerárquico

Existen dos tipos de métodos generales dentro de los catalogados en esta categoría, están aquellos que mezclan grupos para formar uno nuevo (aglomerativos o ascendentes)

y aquellos que separan un grupo existente para dar lugar a dos nuevos (disociativos o descendentes). Estos métodos a su vez, presentan una gran diversidad de variantes.

Haremos énfasis especial en los métodos aglomerativos, pues es en los que se basa el análisis que realizamos en este proyecto. Estas paqueterías están implementadas en diversos softwares estadísticos bajo el nombre de AGNES (AGglomerative NESting).

La idea general de los métodos aglomerativos es la siguiente.

1. Inicie con tantos grupos como puntos haya, donde cada punto va en uno y solo un grupo. La medida de similaridad entre grupos en este paso es igual a la distancia entre los puntos que contiene cada grupo.

2. Encuentre el par de grupos más cercanos (con mayor similitud) y mézclelos en un solo grupo.

3. Calcule las distancias (similaridades) entre el nuevo grupo y cada uno de los grupos antiguos.

4. Repita los pasos 2 y 3 hasta que se alcanze un número deseado de grupos o todos los puntos se hayan mezclado en un solo grupo.

Los métodos jerárquicos nos permiten la constucción de un árbol de clasificación llamado *dendograma* (4.1), el cual nos muestra cuáles grupos se van uniendo y a qué nivel lo hacen, así como la medida de asociación entre los grupos cuando estos se mezclan (nivel de fusión).

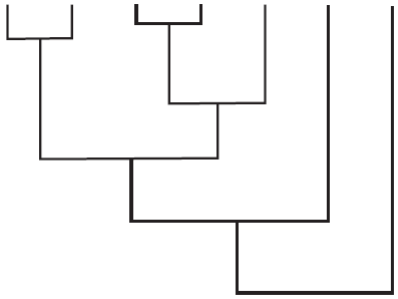


Figura 4.1: Ejemplo de dendograma

Dentro de nuestro particular interés sobre los métodos aglomerativos, explicaremos una de las variantes, los métodos *linkage clustering*.

4.1.1. Distancia mínima o similitud máxima (Single linkage)

En este método se considera que la distancia o similitud entre dos grupos está dada, respectivamente por la distancia mínima (o máxima similitud) entre sus componentes. De este modo, tras efectuar k pasos, tendremos formados $n-k$ grupos, y de esta manera la distancia entre dos grupos G_i (con n_i puntos), G_j (con n_j puntos) sería:

$$d(G_i, G_j) = \min_{x_l \in G_i x_m \in G_j} \{d(x_l, x_m)\}, \qquad l = 1, ..., n_i; \qquad m = 1, ..., n_j. \tag{4.1}$$

Por otro lado, si usamos una medida de similitud entre grupos, tendríamos:

$$s(G_i, G_j) = \underset{x_l \in G_i x_m \in G_j}{\text{máx}} \{s(x_l, x_m)\}, \qquad l = 1, ..., n_i; \qquad m = 1, ..., n_j. \tag{4.2}$$

4.1.2. Distancia máxima o similitud mínima (Complete linkage)

En este método la distancia o similitud se da por los elementos más dispares de dos grupos, es decir, la distancia entre dos grupos está dada por la máxima distancia (o mínima similitud) entre sus componentes. Así pues, como en el caso anterior tras efectuar k pasos tendremos formados $n - k$ grupos, de modo que la distancia entre dos grupos G_i (con n_i puntos) y G_j (con n_j puntos) sería:

$$d(G_i, G_j) = \underset{x_l \in G_i x_m \in G_j}{\text{máx}} \{d(x_l, x_m)\}, \qquad l = 1, ..., n_i; \qquad m = 1, ..., n_j. \tag{4.3}$$

Por otra parte, si usamos una medida de similitud entre grupos, tendríamos:

$$s(G_i, G_j) = \underset{x_l \in G_i x_m \in G_j}{\text{mín}} \{s(x_l, x_m)\}, \qquad l = 1, ..., n_i; \qquad m = 1, ..., n_j. \tag{4.4}$$

4.1.3. Distancia o similitud promedio ponderada (Average distance)

La distancia o similitud entre dos grupos, está definida por el promedio ponderado de las distancias o similitudes de los componentes de un cluster respecto a otro. Sean G_i (con n_i elementos) y G_j (con n_j elementos), supongamos además que G_i está formado por otros dos grupos G_{i_1} y G_{i_2} con n_{i_1} y n_{i_2} elementos respectivamente, de modo que $n_i = n_{i_1} + n_{i_2}$. Así, en términos de distancias (o similitudes), la distancia promedio ponderada es:

$$d(G_i, G_j) = \frac{n_{i_1} d(G_{i_1}, G_j) + n_{i_2} d(G_{i_2}, G_j)}{n_{i_1} + n_{i_2}}. \tag{4.5}$$

4.2. Algoritmo Mapper

La construcción de un complejo simplicial mediante el algoritmo Mapper se da siguiendo algunas ideas de topología que se describen a continuación. Supóngase que se tiene un espacio X y una función continua $f : X \to Z$ a un espacio de referencia Z. Supóngase también que el espacio Z cuenta con una cubierta abierta finita $\mathcal{U} = \{U_\alpha\}_{\alpha \in A}$, A conjunto finito. Dada la continuidad de f, los conjuntos $f^{-1}(U_\alpha)$ forman una cubierta abierta para X. Para cada α, considérese la descomposición de $f^{-1}(U_\alpha)$ en sus componentes conexas, de modo que se pueda escribir $f^{-1}(U_\alpha) = \bigcup_{i=1}^{j_\alpha} V(\alpha, i)$ donde j_α es el número de componentes conexas en $f^{-1}(U_\alpha)$. Denotamos por $\tilde{\mathcal{U}}$ a la cubierta de X obtenida de esta manera.

Para contar con la visualización del espacio a diferentes escalas (multiresolución), se define un mapeo de cubiertas como sigue: dadas dos cubiertas $\mathcal{U} = \{U_\alpha\}_{\alpha \in A}$ y $\mathcal{V} = \{V_\beta\}_{\beta \in B}$ un *mapeo de cubiertas* es una función $f : A \to B$ tal que $U_\alpha \subseteq V_{f(\alpha)}$ para cada $\alpha \in A$.

Ejemplo 2. Singh et al. (2007) Sean $X = [0, N]$ y $\varepsilon > 0$. Los conjuntos $I_l^\varepsilon = (l - \varepsilon, l + \varepsilon + 1) \cap X$, para $l = 0, 1, ..., N - 1$ forman una cubierta abierta $\mathcal{I}_\varepsilon$ para X. Todas las cubiertas $\mathcal{I}_\varepsilon$ tienen el mismo conjunto de índices, y para $\varepsilon \leqslant \varepsilon'$, el mapeo identidad en el conjunto de índices es un mapeo de cubiertas dado que $I_l^\varepsilon \subset I_l^\varepsilon$.

Recordemos que para una cubierta $\mathcal{U}$ de un espacio X, $N(\mathcal{U})$ detona el nervio de $\mathcal{U}$. Si tenemos dos cubiertas $\mathcal{U}$ y $\mathcal{V}$ y un mapeo de cubiertas f, entonces existe un mapeo inducido de complejos simpliciales $N(f) : N(\mathcal{U}) \to N(\mathcal{V})$, dado sobre los vértices por el mapeo f. En consecuencia, si tenemos una familia de cubiertas $\{\mathcal{U}_i\}_{i \in n}$ y mapeos de cubiertas $f_i : \mathcal{U}_i \to \mathcal{U}_{i+1}$ para cada i, obtenemos un diagrama de complejos simpliciales y mapeos simpliciales

$$N(\mathcal{U}_0) \overset{N(f_0)}{\longrightarrow} N(\mathcal{U}_1) \overset{N(f_1)}{\longrightarrow} \cdots \overset{N(f_{n-1})}{\longrightarrow} N(\mathcal{U}_N).$$

Retomando el caso del espacio X y la función $f : X \to Z$, y un mapeo de cubiertas $\mathcal{U} \to \mathcal{V}$, existe el correspondiente mapeo de cubiertas $\bar{\mathcal{U}} \to \bar{\mathcal{V}}$.

Para pasar de la idea teórica a la implementación se hace uso de herramientas estadísticas. De manera particular, se utilizan las técnicas de agrupamiento jerárquico descritas en la Sección 4.1. El objetivo es realizar una partición del espacio de interés en componentes conexas. El algoritmo para construir un complejo simplicial es el siguiente:

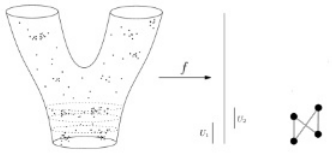
1. Se encuentra el rango I de la función restringida a la nube de puntos.

2. Se particiona I en un conjunto $\mathcal{I} = \{I_1, I_2, ..., I_k\}$ para un k deseado. A los intervalos se les permite traslaparse un porcentaje p. Se puede controlar el traslape p, así como la longitud de los intervalos de $\mathcal{I}$.

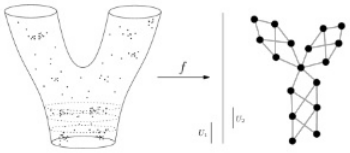
3. Para cada intervalo $I_j \in \mathcal{I}$, se encuentra el conjunto de puntos $X_j = \{x | f(x) \in I_j\}$. De este modo, la familia de conjuntos $\{X_j\}$ forman una cubierta de X.

4. Para cada conjunto X_j se encuentran grupos $\{X_{jk}\}$. De modo que se tratará cada grupo como un vértice en nuestro complejo y dibujamos una arista entre los vértices siempre que $X_{jk} \cap X_{lm} \neq \emptyset$.

Se ilustra este algoritmo en la siguiente figura.



(a) (a)



(b) (b)

Figura 4.2: La figura (a) muestra la construcción de un 1-simplejo entre dos intervalos en el rango del filtro f. La figura (b) ilustra la construcción completa dada una cubierta (con traslape p) del rango de la función f. (Bak, 2016)

Si deseamos obtener información sobre características de agujeros de dimensiones mayores, es necesario construir un complejo simplicial de dimensión mayor. Para esto es necesario utilizar tantas funciones (en adelante *filtros*) como dimensiones se tengan. Cualquier cubierta del espacio de referencia podría funcionar, pero se debe tener en cuenta que cuanto más intersecciones haya en dicha cubierta, más complejos simpliciales de dimensión mayor habrá (ver Figura 4.3).

Para ilustrar la construcción de un complejo simplicial de dimensión mayor, Singh et al. (2007) consideran como caso particular una nube de datos en $\mathbb{R}^2$. Se necesitan dos funciones filtro f_1, f_2 y se supone además que el rango de (f_1, f_2) queda cubierto por rectángulos. Tenemos la región $\mathcal{R} = [\text{mín}\, f_1, \text{máx}\, f_1] \times [\text{mín}\, f_2, \text{máx}\, f_2]$. De este modo se tiene una cubierta de $\mathcal{R}$ tal que cada $A_{i,j}, A_{i+1,j}$ se intersectan al igual que cada $A_{i,j}, A_{i,j+1}$. El algoritmo para calcular un complejo simplicial reducido es el siguiente:

1. Para cada i, j, se eligen los puntos para los cuales los valores de las funciones f_1, f_2 caen en $A_{i,j}$. Se encuentran los grupos para este conjunto y considere que cada grupo representa un vértice ($0-$simplejo). Manténgase una lista de vértices para cada $A_{i,j}$ y un conjunto de índices para los puntos de cada grupo.

2. Para todos los vértices en los conjuntos $\{A_{i,j}, A_{i+1,j}, A_{i,j+1}, A_{i+1,j+1}\}$, si la intersección de los grupos asociados con los vértices es no vacía añadimos una arista ($1-$simplejo).

3. Cuando los grupos correspondientes a cualesquiera 3 vértices tengan intersección no vacía, añadimos un triángulo (2−simplejo) con esos 3 vértices.

4. Cuando los grupos correspondientes a cualesquiera 4 vértices tengan intersección no vacía, añadimos un tetraedro (3−simplejo) con esos 4 vértices.

Siguiendo esta misma idea, se puede extender Mapper a un espacio de dimensión mayor.

Existe un problema cuando se utiliza una cubierta del espacio por rectángulos, pues es posible que se intersecten hasta 4 de ellos (Figura 4.3). Si seguimos el algoritmo descrito anteriormente, se pueden generar simplejos de hasta dimensión 4. Es posible intercambiar la cubierta de rectángulos por una cubierta de hexágonos, de modo que sea posible controlar mediante un parámetro el tamaño de los hexágonos y cómo estos intersectan. De este modo se puede regular el parámetro para que se intersecten a lo más 3 hexágonos, creando así simplejos de dimensión a lo más 3.

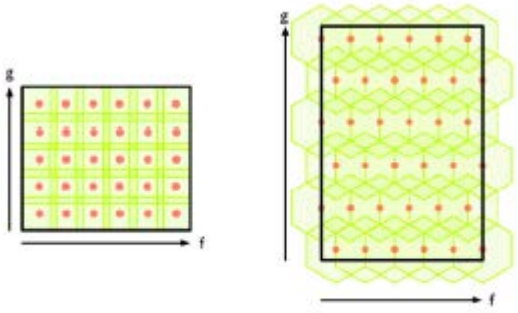


Figura 4.3: Ilustración de cubierta de $\mathbb{R}^2$ por rectángulos (izquierda) y hexágonos (derecha). (Singh et al., 2007)

4.2.1. Filtros

El algoritmo Mapper es altamente dependiente de los filtros que se eligen para particionar el conjunto de datos. Se asume que la nube de puntos está dotada de una función distancia $d(x, y)$, se mencionan a continuación algunas de las funciones utilizadas en Mapper que describen algunas propiedades estructurales de los datos. A partir de aquí estamos considerando que X es una nube de datos y el espacio de referencia Z es el de los números reales. Describimos algunos filtros en las siguientes secciones.

4.2.1.1. Exentricidad

La idea intuitiva es encontrar los puntos que se encuentren alejados de un "centro". Dado $1 \leqslant p \leqslant +\infty$,

$$E_p(x) = \left(\frac{\sum_{y \in X} d(x,y)^p}{N} \right)^{\frac{1}{p}}, \text{ con } x, y \in X.$$

Se puede extender la definición a $p = +\infty$ haciendo $E_\infty(x) = \text{máx}_{x' \in X}\, d(x, x')$. En general, tiende a tomar valores grandes para puntos que estan alejados de un "centro".

4.2.1.2. Laplacianos de grafos

Esta familia de funciones se origina de considerar el operador Laplaciano de un grafo definido como sigue: El conjunto de vértices de este grafo es el conjunto X de todos los puntos, y el peso de las aristas entre los puntos $x, y \in X$ es

$$w(x, y) = k(d(x, y)),$$

donde k es un kernel de suavizamiento. Una matriz Laplaciana del grafo (normalizada) se calcula como

$$L(x, y) = \frac{w(x, y)}{\sqrt{\sum_z w(x, z)}\sqrt{\sum_z w(y, z)}}.$$

De este modo, los eigenvectores de la matriz Laplaciana normalizada del grafo nos dan un conjunto de vectores ortogonales que nos brindan información geométrica interesante de la nube de datos.

4.2.1.3. Componentes de la SVD

Dada una matriz de datos podemos aplicarle *descomposición en valores singulares* (SVD por sus siglas en inglés) para obtener el $k-$ésimo valor propio de una matriz de distancias, por ejemplo el vector propio principal corresponde al valor propio de mayor magnitud. Al proyectar los datos en dicho vector estamos realizando una reducción de dimensionalidad, tal proyección nos puede servir como un filtro y podemos usarla para obtener características topológicas de una nube de puntos. Las proyecciones en otros vectores propios posiblemente podrían darnos una visión distinta al vector propio principal.

Definición 42 (Descomposición en Valores Singulares (SVD), (Demmel, 1997))**.** Sea A una matriz arbitraria de tamaño $m \times n$ con $m \geqslant n$. Podemos escribir $A = U\Sigma V^T$, donde U es de tamaño $m \times n$ y satisface que $U^T U = I$, V es de tamaño $n \times n$ y satisface que $V^T V = I$, y $\Sigma = \text{diag}(\sigma_1, ..., \sigma_n)$, donde $\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_n \geqslant 0$. Las columnas $u_1, ..., u_n$ de U se llaman vectores singulares izquierdos. Las columnas $v_1, ..., v_n$ de V se llaman vectores singulares derechos. Los σ_i se llaman valores singulares. (Si $m < n$, definimos la SVD considerando A^T).

4.2.1.4. Estimador vía kernel

También es posible utilizar como filtro el estimador de densidades vía kernel descrito en la Sección 2.5.

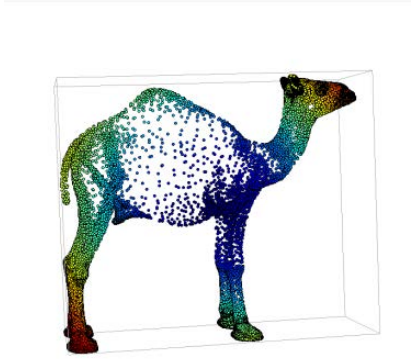
4.3. Software

En la red se pueden encontrar un par de implementaciones de Mapper desarrolladas en lenguaje Python. Tanto la instalación como el código de ambas son de libre distribución, en las referencias de cada autor se encuentran las ligas a sus sitios web desde donde pueden seguirse instrucciones para la instalación del software.

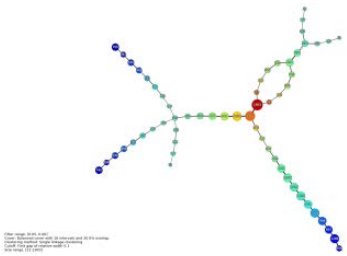
4.3.1. Python Mapper

Una primera implementación en Python de este algoritmo, fue desarrollada por Müllner y Babu (2013). Los resultados que entrega el software hasta ahora solamente son grafos no dirigidos codificados con colores que varían de acuerdo al filtro seleccionado así como el tamaño de los vértices dependiendo de la densidad de puntos en vecindades cercanas. Los filtros que se pueden trabajar son los 3 mencionados en la Sección 4.2.1.

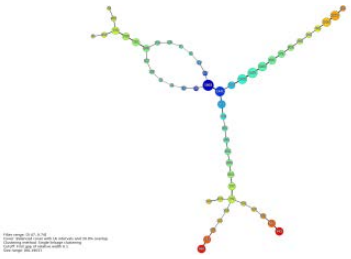
El ejemplo que presentamos a continuación se realizó utilizando los 3 filtros descritos en la sección anterior. Comparamos en una misma figura los 3 distintos filtros para ilustrar que los resultados de estos son similares ajustando los parámetros de cada filtro.



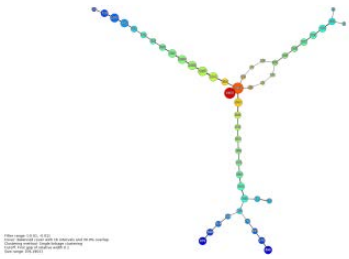
(a) 21887 puntos de una figura 3D de un camello.



(b) Grafo Kernel con ancho de banda $h = 1$.



(c) Grafo Exentricidad parámetro $p = 1$



(d) Grafo SVD, primera componente.

Figura 4.4: Uso del algoritmo Mapper con 3 filtros distintos.

El ejemplo de la Figura 4.5 tiene como objetivo ilustrar la interpretación del grafo que nos entrega el algoritmo. Usamos como filtro la primer componente de la SVD, con 25 intervalos y permitiendo que éstos se traslapen hasta un 30 %. El algoritmo de agrupamiento utilizado es el de distancia mínima, con un parámetro de corte de 0.1.

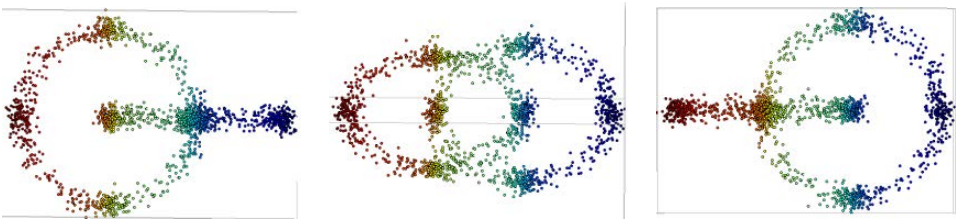


Figura 4.5: Distintas perspectivas de una nube datos sobre un doble anillo cruzado sin intersección. Las entradas de cada círculo fueron perturbadas mediante un proceso inverso gaussiano de parámetros $\mu = 0.1$ y $\lambda = 0.0001$.

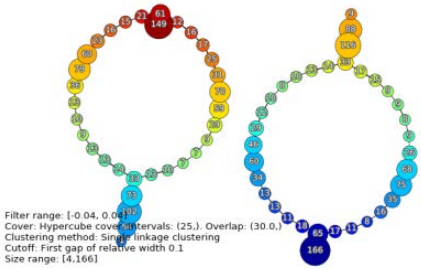


Figura 4.6: Grafo Mapper de la nube de datos sobre un doble anillo.

En la Figura 4.6 podemos observar la relación que guardan los colores de los dos grafos con los dos anillos en 4.5. Al tener dos círculos separados nos indica que se trata de dos componentes conexas, donde cada una tiene un agujero de dimensión 1. El tamaño de los vértices del grafo (así como el número) nos indica mayor concentración en esas zonas de la nube de puntos. Esto último lo ilustramos en la Figura 4.7.

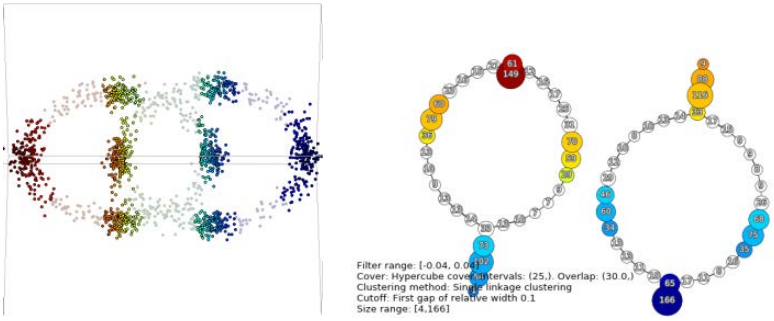


Figura 4.7: Selección en Mapper de los puntos de concentración de los datos en un doble anillo.

En esta última figura, se puede notar cómo el algoritmo detecta los 8 puntos de concentración generados por las perturbaciones a la distribución uniforme en el doble anillo mediante el proceso inverso gaussiano.

4.3.2. Kepler Mapper

Otra implementación desarrollada por van Veen (2015) también en Python se encuentra disponible en su sitio de GitHub. El software consta solamente de un módulo en Python que debemos colocar en una carpeta que nuestro compilador reconozca. Este modulo toma como base los algoritmos de agrupamiento y escalamiento incluidos en el paquete Scikit-Learn utilizado en aprendizaje de máquinas. Los resultados se interpretan de la misma manera que Python Mapper. A pesar de que ambas implementaciones se distribuyen con permisos para la modificación del código, la implementación realizada por van Veen nos permite modificar de manera más clara sus líneas de modo que uno tiene control sobre lo que el programa está haciendo.

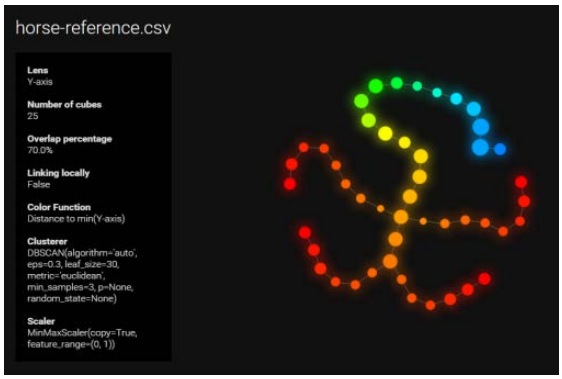


Figura 4.8: Estructura geométrica de una figura con forma de caballo en 3D obtenida mediante Kepler Mapper (van Veen, 2015).

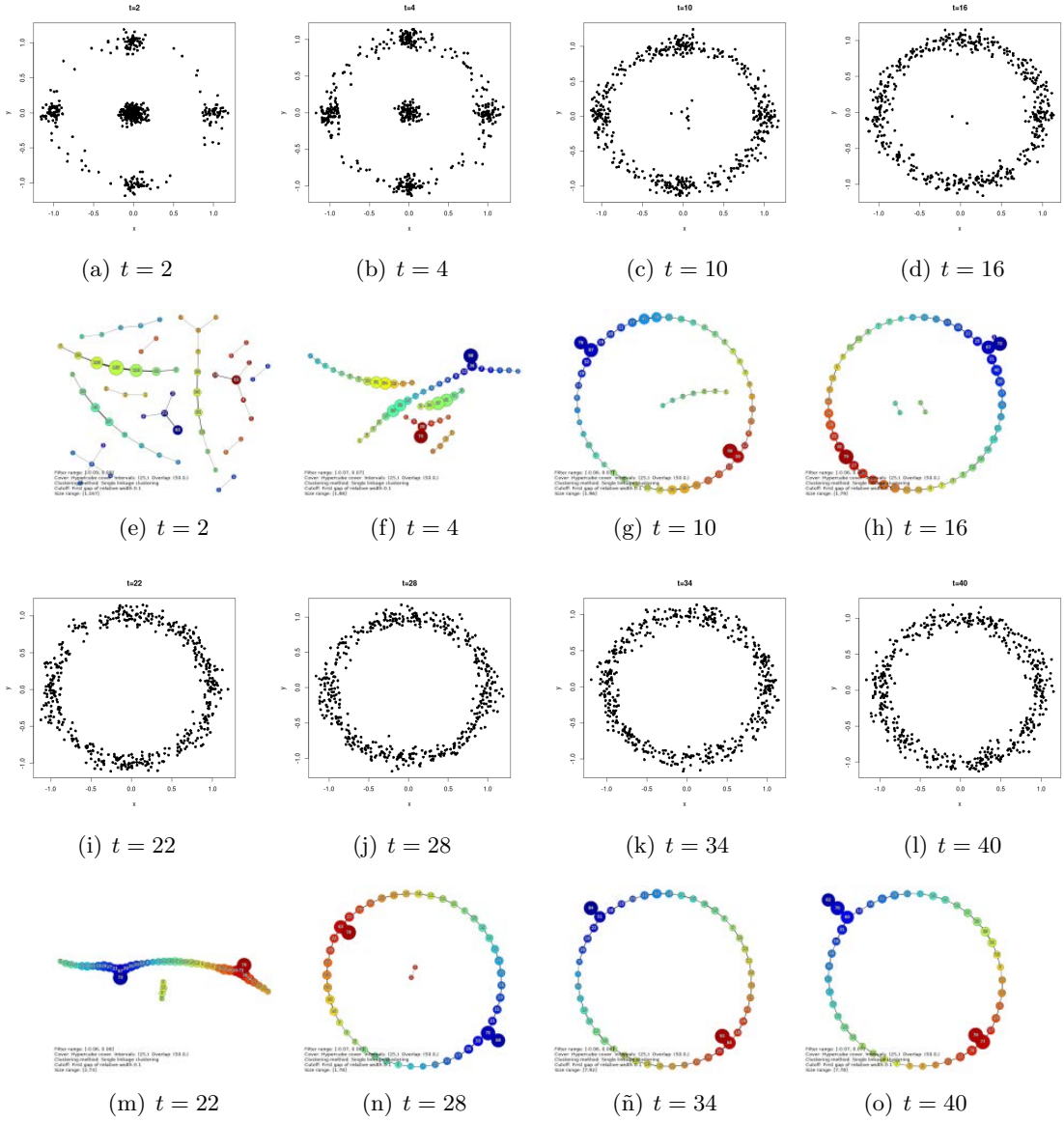
Figura 4.9: Estructura geométrica de una figura con forma de caballo en 3D obtenida mediante Kepler Mapper (van Veen, 2015).

4.4. Ejemplos

En los siguientes ejemplos, las nubes de datos utilizadas son las generadas en el Capítulo 2. Como se mencionó en la Sección 4.3.1, las implementaciones disponibles de este algoritmo devuelven un grafo coloreado, que sólo nos permite ver el número de componentes conexas y los agujeros 1−dimensionales presentes en la nube de datos. Hacemos énfasis en

que Mapper nos ayuda sobre todo a encontrar la estructura geométrica de nuestras nubes de puntos al detectar cuántas componentes conexas existen, así como la manera en que los puntos están distribuidos a lo largo de dicha estructura.

El filtro utilizado en los ejemplos es la primer componente de la SVD, con un traslape de 50 % en los intervalos de la cubierta. Se utiliza el método de agrupamiento por distancia mínima.



(a) $t = 2$ (b) $t = 4$ (c) $t = 10$ (d) $t = 16$

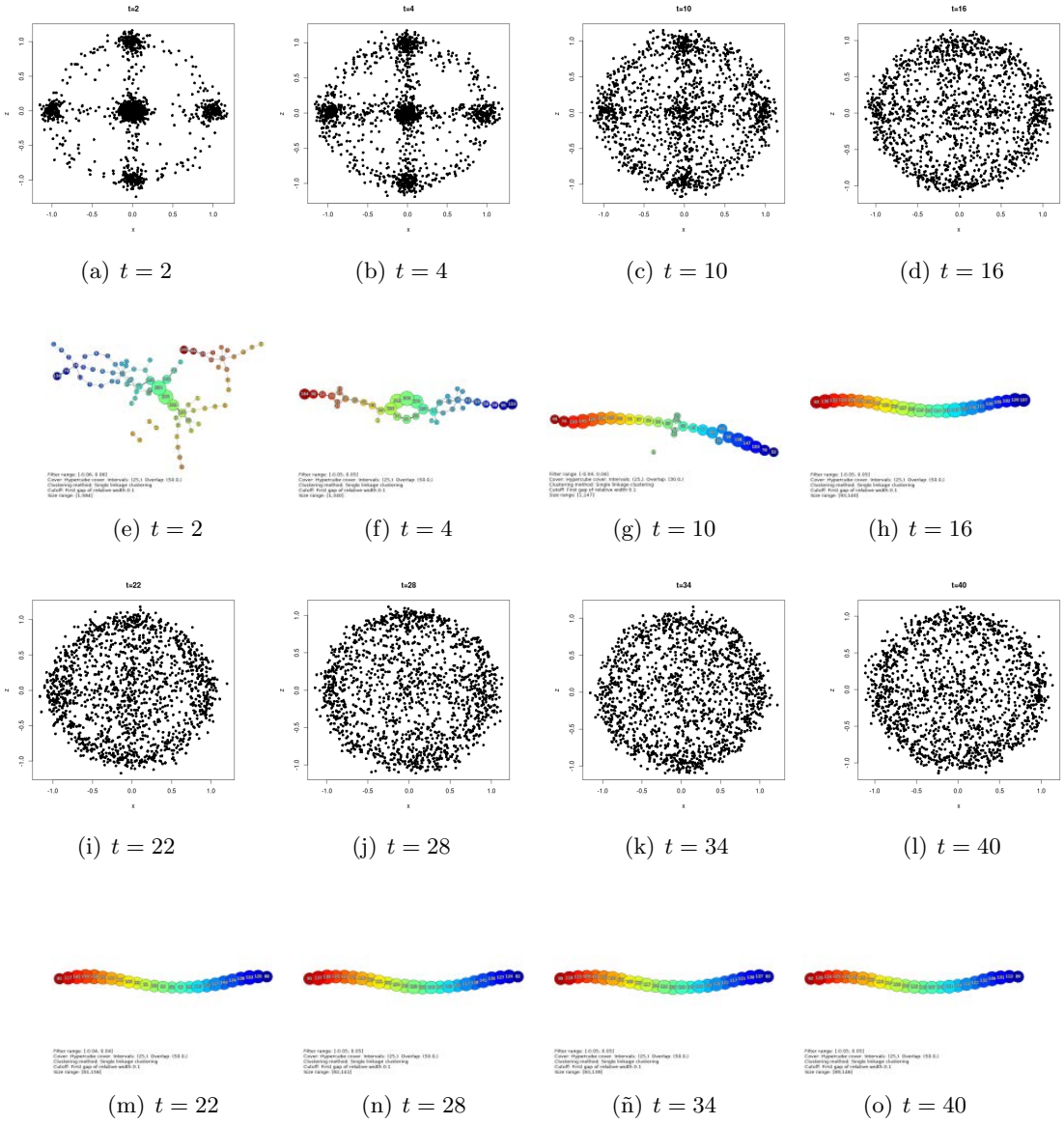
(e) $t = 2$ (f) $t = 4$ (g) $t = 10$ (h) $t = 16$

(i) $t = 22$ (j) $t = 28$ (k) $t = 34$ (l) $t = 40$

(m) $t = 22$ (n) $t = 28$ (ñ) $t = 34$ (o) $t = 40$

Figura 4.10: 500 datos sobre $\mathbb{S}^1$ con coeficientes de perturbación mediante procesos de Poisson de parámetro $\lambda = 0.2$.

Se puede observar en la Figura 4.10 que para los tiempos $t = 2,\ 4$, el algoritmo detecta como componentes conexas cada concentración de puntos (en total 5). Para los tiempos $t = 10,\ 16$, podemos ver al centro del grafo Mapper vértices que representan a los puntos al centro de la nube de datos. El círculo cerrado indica la presencia de un agujero de dimensión 1. En este mismo tiempo se capturan las concentraciones en los ejes cartesianos (vértices de mayor tamaño). Para el tiempo $t = 22$, el algoritmo deja de detectar el agujero $1-$dimensional, pues si observamos en la esquina superior izquierda, se ve una especie de ruptura en la nube de datos. A partir del tiempo $t = 28$ se logra capturar una componente conexa y un agujero $1-$dimensional característicos de $\mathbb{S}^1$.



(a) $t = 2$ (b) $t = 4$ (c) $t = 10$ (d) $t = 16$

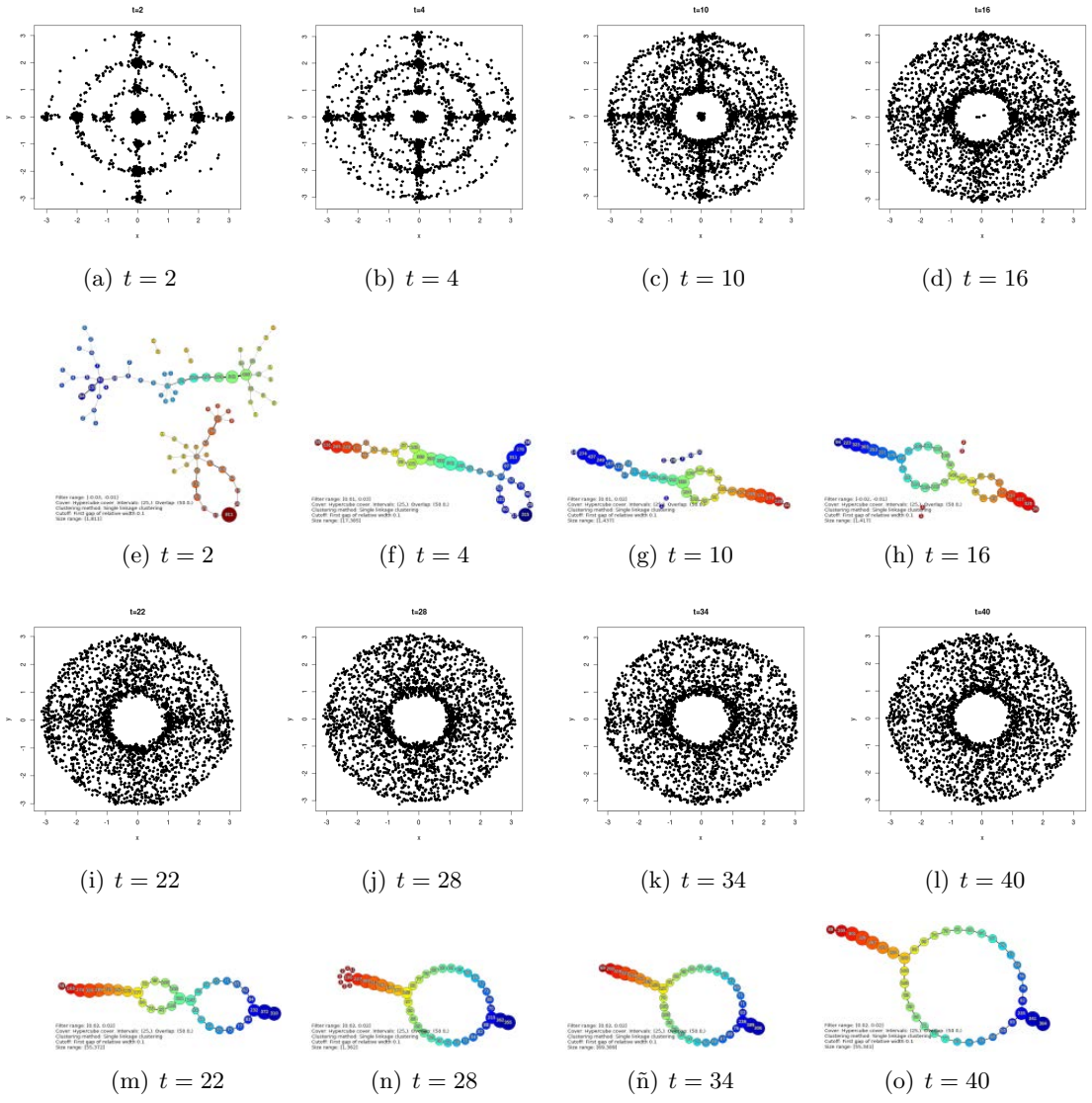
(e) $t = 2$ (f) $t = 4$ (g) $t = 10$ (h) $t = 16$

(i) $t = 22$ (j) $t = 28$ (k) $t = 34$ (l) $t = 40$

(m) $t = 22$ (n) $t = 28$ (ñ) $t = 34$ (o) $t = 40$

Figura 4.11: 1500 datos sobre $\mathbb{S}^2$ con coeficientes de perturbación mediante procesos de Poisson de parámetro $\lambda = 0.2$.

Se puede ver en la Figura 4.11 que para el tiempo $t = 2$ el algoritmo detecta dos componentes conexas. En cada componente conexa podemos ver ramificaciones a partir de los puntos de mayor concentración, éstas son las "lineas" que unen a cada punto de concentración. Para $t = 4,\ 10$ se pueden observar varios ciclos a lo largo del recorrido (agujeros $1-$dimensionales). A partir de $t = 16$ el algoritmo nos indica que se encuentra una sola componente conexa y dejó de encontrar agujeros de dimensión 1. Recordemos que el algoritmo disponible no puede detectar agujeros de dimensión mayor a 1, y es por esto que no detecta el número de Betti $\beta_2 = 1$.



(a) $t = 2$ (b) $t = 4$ (c) $t = 10$ (d) $t = 16$

(e) $t = 2$ (f) $t = 4$ (g) $t = 10$ (h) $t = 16$

(i) $t = 22$ (j) $t = 28$ (k) $t = 34$ (l) $t = 40$

(m) $t = 22$ (n) $t = 28$ (ñ) $t = 34$ (o) $t = 40$

Figura 4.12: 2500 datos sobre $\mathbb{T}^2$ con coeficientes de perturbación mediante procesos de Poisson de parámetro $\lambda = 0.2$.

En la Figura 4.12 podemos ver que para el tiempo $t = 2$ el algoritmo detecta 2 componentes conexas, y cada componente conexa tiene al menos 4 puntos con alta concentración

de datos. Las ramificaciones que salen de cada componente resultan ser las lineas de puntos que notamos unen a los ejes principales que se tienen en el toro. Para los tiempos $t = 4,\ 10,\ 16,\ 22$ podemos ver que el algoritmo encuentra dos ciclos en el recorrido del plano xy. Es a partir del tiempo $t = 28$ cuando se encuentra de manera correcta el grafo, como el que mostramos en la Figura 4.13. El algoritmo no es capaz de captar agujeros de dimensión 2. Además, para figuras en $\mathbb{R}^3$ el filtro recorre el plano xy. Se obtiene la siguiente estructura en el grafo Mapper para $\mathbb{T}^2$.

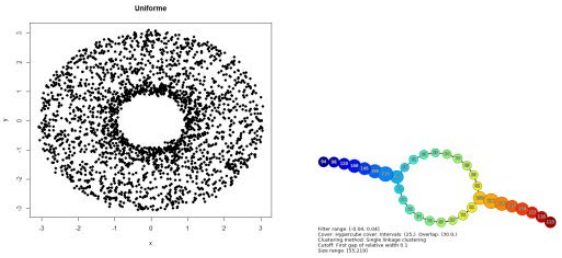


Figura 4.13: 2500 datos con distribución uniforme en $\mathbb{T}^2$.

4.5. Complejo mediante agrupamiento

Los complejos mediante agrupamiento surgen con el objetivo de reducir el costo computacional en la construcción de los complejos clásicos. La característica principal de este tipo de complejos es que la complejidad computacional escala de buena manera con el tamaño de la muestra ($O(n \log n)$) y al mismo tiempo con la dimensión del espacio ambiente ($O(d \log d)$). Esta construcción fue propuesta por Hennigan (2015), presentamos la idea general del autor pues consideramos que es un método que propone una solución eficiente en cuanto a la complejidad computacional.

La técnica propuesta hace uso de un método llamado potencia iterativa, el cual calcula el vector propio correspondiente al mayor valor propio de la matriz de distancias de una nube de datos dada. La ventaja de este método es que solamente calcula ese vector y se evita el cálculo completo de la descomposición SVD. También se utiliza una aproximación del cálculo del método de agrupamiento de distancia mínima, presentado en la Sección 4.1. La idea de esta aproximación es encontrar los centroides de dos grupos A y B para luego proyectar ambos grupos en la línea que los une.

Sea $X_{n\times d}$ una nube de datos de forma matricial, donde cada renglón indica una observación y el número de columnas es la dimensión de la misma. Se define Δ_0 como

$$\Delta_0 = \text{media}\left\{ \|x_i - \frac{1}{n}\sum_{j=1}^{n} x_j\| : x_i \text{ renglón de } X \right\}.$$

Por otro lado, la aproximación del cálculo del agrupamiento mediante distancia mínima para dos grupos A y B se realiza como sigue:

1. Seleccione la mitad de los puntos del grupo A que estén más cercanos al grupo B, el resto se descarta. Denotemos esta selección por A_1.

2. Seleccione la mitad de los puntos del grupo B que estén más cercanos al grupo A, el resto se descarta. Denotemos esta selección por B_1.

3. Repítase 1 y 2 hasta que sólo quede un punto en cada grupo A_k, B_k para algún k.

   Se denota como $\Delta_1(A, B)$ a la distancia $d(A_k, B_k)$.

Nota: Se utiliza la distancia Manhattan para este cálculo. Es decir, para dos puntos $x, y \in \mathbb{R}^d$ se tiene que $d(x, y) = \sum_{i=1}^{d} |x_i - y_i|$.

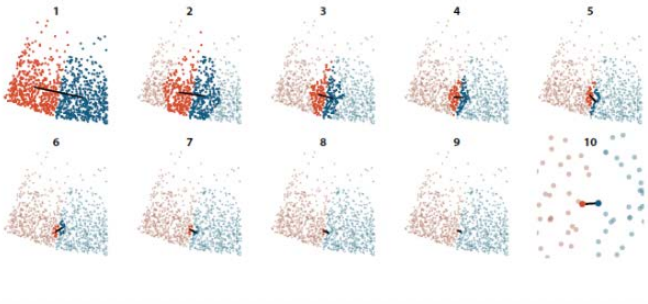


Figura 4.14: Cálculo de la distancia $\Delta_1(A, B)$ (Hennigan, 2015)

Se puede generalizar la idea del cálculo de $\Delta_1(A, B)$ para k grupos $A_1, A_2, ..., A_k$. $\Delta_k(A_1, ..., A_k)$ es el volumen (en dimensión k) del simplejo "más pequeño" generado con un punto en cada grupo. Nos referimos a "más pequeño" en el sentido que la distancia entre cada grupo sea mínima.

Así, se crea el *complejo mediante agrupamiento* con escala de multiresolución t como sigue:

1. Calcule el eje principal p_x mediante potencia iterativa.

2. Proyecte los puntos de X sobre p_x para obtener u, cada entrada de u representa a una observación de X.

3. Los puntos que se proyecten a un valor menor que la mediana de u se colocan en X_L, y el resto van a X_R.

4. De manera recursiva, construya un árbol de ejes principales en X_L y X_R a la profundidad deseada.

5. Se añade la arista $[\sigma_i \sigma_j]$ si

$$\text{máx}(\Delta_0(X_i), \Delta_0(X_j)) + \Delta_1(X_i, X_j) \leqslant t.$$

6. De manera inductiva, se añade el $k-$simplejo si

$$\underset{i \in k}{\text{máx}} \left\{ \Delta_{k-1}(X_1, ..., \hat{X}_i, ..., X_k) \right\} + \Delta_k(X_1, ..., X_k) \leqslant t.$$

5

Análisis de costo computacional, robusticidad y eficiencia

El objetivo de este capítulo es presentar los resultados de estudios de simulación comparativos entre los algoritmos simpliciales descritos en los Capítulos 2, 3 y 4, con relación al costo computacional y a la capacidad con respecto al tamaño de muestra de las nubes de puntos y la dimensión de la variedad.

En la Sección 5.1 se mencionan los resultados de un estudio comparativo reportado en la literatura de ATD sobre las implementaciones en distintos lenguajes de los algoritmos Čech, VR y alfa. En la Sección 5.2 presentamos los resultados de la comparación de los algoritmos VR, MS, alfa y testigo mediante un análisis de costo computacional, el cual incluye tiempo de ejecución y cantidad de memoria física y virtual utilizada. Por último, en la Sección 5.3 se presentan resúmenes de los resultados obtenidos mediante simulación donde se comparan la distancia cuello de botella entre diagramas de persistencia y la distancia Hausdorff entre los conjuntos de datos. Esto tiene la finalidad de observar qué tan conservadora es la desigualdad en el teorema de estabilidad introducido en el Capítulo 2.

5.1. Costo computacional: datos reales

Otter, Porter, Tillmann, Grindrod y Harrington (2015) realizan un estudio comparativo entre las implementaciones en distintos lenguajes de los algoritmos para calcular la homología simplicial a través de las filtraciones de VR, Čech y alfa. En primera instancia se tiene el tamaño de los complejos simpliciales, el cual depende del orden de complejidad computacional de cada algoritmo. En la siguiente tabla se muestran posibles aproximaciones y/o reducciones para hacer menos costoso el tiempo de ejecución en cada caso.

Complejo $\mathcal{K}$	Tamaño de $\mathcal{K}$	(A) Aproximación	(B) Reducción
Čech	$2^{\mathcal{O}(n)}$	-	Conjunto ordenado, red. Morse
Vietoris-Rips	$2^{\mathcal{O}(n)}$	Lineal, testigo,GIC,HOPES	Conjunto ordenado, red. Morse
α	$n^{\mathcal{O}(\lfloor d/2 \rfloor)}$	HOPES (1D)	Conjunto ordenado, red. Morse

Tabla 5.1: Tabla del tamaño de los complejos simpliciales "usuales", sus posibles aproximaciones y/o reducciones (Otter et al., 2015).

Los autores realizan un análisis de diversas paqueterías de software disponibles mediante distribución libre. En la página personal de Nina Otter hay un compendio de los distintos programas disponibles (`https://people.maths.ox.ac.uk/otter/PH_programs`). Se utilizan 6 conjuntos de datos, los primeros dos son simulados y el resto son datos reales:

1. Botella de Klein: Usan muestras de 400 y 900 puntos de la inmersión de la botella de Klein en $\mathbb{R}^3$.

2. Complejos aleatorios VR: Se crean complejos $V(X, \varepsilon)$ donde $\text{card}(X) = n$. Con $n = 100, 1000$).

3. Secuencias genómicas del Virus de VIH.

4. Gráfica del Dragón de Stanford: Es una reconstrucción geométrica de una captura 3D del dragón.

5. Red Neuronal *C. elegans*.

6. Genoma humano.

   Para mayor detalle ver Otter et al. (2015).

Los autores examinan las paqueterías Javaplex, Perseus, Dionysus, DIPHA y GUDHI usando los conjuntos de datos arriba descritos. Los criterios de medición que utilizan son:

1. Rendimiento en segundos de CPU y tiempo transcurrido.

2. Memoria utilizada por el proceso.

3. Número máximo de complejos simpliciales permitidos por el software.

4. Fases de cálculo de la homología persistente soportadas por el software.

El equipo de cómputo en el que se llevó a cabo el desarrollo computacional del trabajo tiene las siguientes características:

- 1728 núcleos (108x16) a 2.0GHz Xeon SandyBridge.

- 64GB de RAM en 80 nodos y 128GB en 4 nodos.

- 20TB de almacenamiento.

En las Tablas 5.2, 5.3, 5.4 y 5.5 se muestran las tablas que resumen el rendimiento, el uso de memoria, el tamaño máximo de los complejos simpliciales soportado por las librerías y los pasos disponibles en cada paquete.

Data set	C. elegans		Klein		HIV		Dragon 1		Dragon 2	
size of complex	4.4×10^6		1.1×10^7		2.1×10^8		1.7×10^8		1.3×10^9	
javaPlex st	84	284	747	1031	-	-	-	-	-	-
Dionysus st	474	473	1830	1824	-	-	-	-	-	-
DIPHA st	6	68	90	1360	1631	25950	7356	117417	142559	1489615
Perseus	543	542	1978	1974	-	-	-	-	-	-
javaPlex d **										
Dionysus d	513	513	145	145	-	-	4360	4362	-	-
DIPHA d	4	39	6	73	81	1276	75	1176	2358	37572
GUDHI	6	4	11	10	249	248	285	283	15419	3151

Tabla 5.2: Rendimiento de las paqueterías en tiempo de CPU y tiempo transcurrido en segundos para los conjuntos *C. elegans*, botella de Klein, VIH y dragón de Stanford. Para cada conjunto, se indica el tamaño del complejo simplicial. 'st' después del nombre del paquete indica la implementación del algoritmo estándar y la implementación del algoritmo dual se etiqueta con 'd'. En Perseus sólo se implementa el algoritmo estándar y GUDHI implementa solamente el algoritmo dual. Se corrió DIPHA en un nodo y 16 núcleos para los conjuntos *C. elegans*, botella de Klein y el dragón 1; el conjunto de VIH se corrió en 2 nodos de 16 núcleos y el conjunto Dragón 2 en 2 y 3 nodos de 16 núcleos para la implementación dual y estándar respectivamente (tabla extraída de Otter et al. (2015)).

Data set	C. elegans	Klein	HIV	Dragon 1	Dragon 2
size of complex	4.4×10^6	1.1×10^7	2.1×10^8	1.7×10^8	1.3×10^9
javaPlex st	< 5	< 15	> 120	> 120	> 120
Dionysus st	1.3	11.6	-	-	-
DIPHA st	0.1	0.2	2.7	2.4	4.9
Perseus	5.1	12.7	-	-	-
javaPlex d					
Dionysus d	0.5	1.1	-	16.8	-
DIPHA d	0.1	0.2	1.8	1.8	13.8
GUDHI	0.2	0.6	9.9	9.2	64.5

Tabla 5.3: Uso de memoria en GB para los mismos conjuntos de la Tabla 5.2. En JavaPlex se indica la pila máxima que fue necesaria para realizar el cálculo; el valor indicado es un límite superior para el uso de memoria. Para DIPHA, se indica la cantidad máxima de memoria utilizada por un solo núcleo (considerando todos). La diferencia en el uso de memoria entre la implementación dual y estándar del algoritmo DIPHA en el caso del conjunto Dragon 2, se usaron 32 núcleos para la dual y 48 para la estándar. (tabla extraída de Otter et al. (2015)).

Software	javaPlex		Dionysus		DIPHA		Perseus	GUDHI
	st	d	st	d	st	d	st	d
max. size	$1 \cdot 10^7$		$1.1 \cdot 10^7$	$3.2 \cdot 10^8$	$1.3 \cdot 10^9$	$1.3 \cdot 10^9$	$1 \cdot 10^7$	$1.3 \cdot 10^9$

Tabla 5.4: Tamaño máximo soportado de los complejos simpliciales para cada paquete (tabla extraída de Otter et al. (2015)).

Software	javaPlex	Perseus	Dionysus	DIPHA	GUDHI
Installation	✓	✓	–	–	–
Complex	✓	✓	✓	✓	✓
Boundary matrix	✓	✓	✓	✓	✓
Barcodes	✓	✓	✓	✓	✓
Visualisation	✓	✓	–	✓	–

Tabla 5.5: Pasos implementados por las distintas librerías: (1) Instalación del software, (2) cálculo del complejo a partir de datos, (3) cálculo de la matriz de fronteras, (4) cálculo de códigos de barra y (5) visualización de las salidas. Se indica qué pasos están soportados por cada software (tabla extraída de Otter et al. (2015)).

De la Tabla 5.3 podemos ver que con los recursos con que contamos para el desarrollo de este trabajo de tesis puede ser posible llevar a cabo el cálculo de la homología para complejos simpliciales grandes (aunque es necesario saber cuántos GB más implica el símbolo > 120 en la tabla). A pesar de contar con un gran recurso en el equipo que se desarrolló el artículo, se puede observar que el algoritmo DIPHA tarda un tiempo considerable en el caso de la botella de Klein (1360 segundos para 900 puntos en $\mathbb{R}^3$). De todos los algoritmos, el más rápido en cuanto a tiempo de ejecución es el algoritmo GUDHI.

La paquetería GUDHI fue de nuestro particular interés, pues se tienen diversas implementaciones de algoritmos de ATD (árboles simpliciales, complejos alfa, entre otras). Sin embargo, la información de la Tabla 5.5 nos ayudó a dilucidar el por qué no nos fue posible instalar esta paquetería. Esto se debe a que no se tiene suficiente información para su correcta instalación. También intentamos instalar la librería Dionysus sin éxito, misma que no se encuentra marcada en la Tabla 5.5.

Sin embargo, las paqueterías GUDHI, Dionysus y DIPHA se encuentran implementadas en la paquetería TDA de R que se explican Fasy et al. (2014). Aunque en este caso, la paquetería es poco eficiente cuando se usa en equipos de escritorio, incluso en el servidor que utilizamos hubo momentos en que el algoritmo no podía calcular la homología de distribuciones que presentaban huecos en las variedades encajadas en $\mathbb{R}^3$.

Una buena alternativa a probar sería el algoritmo DIPHA pues en la Tabla 5.3 podemos observar que para todos los casos el uso de memoria no excedió los 15GB. Sin embargo, es necesario señalar que en la Tabla 5.2 se ve que el tiempo de ejecución es mucho mayor que en otros casos.

5.2. Costo computacional: simulación

En esta sección presentamos un estudio de simulación cuyo objetivo es ampliar lo mostrado por Otter et al. (2015). Esto último en el sentido que trabajamos datos simulados en las variedades $\mathbb{S}^1$, $\mathbb{S}^2$ y $\mathbb{T}^2$ con las distribuciones que se presentaron en el Capítulo 2, utilizando modelos de probabilidad concretos. Más aún, se modifica la cantidad de ruido que se añade a las nubes de datos simuladas y se trabaja con distintos tamaños de muestra con el propósito de analizar los cambios en el costo computacional y la eficiencia de los algoritmos al calcular la topología subyacente a los datos.

En el estudio de simulación se incluyen los algoritmos Vietoris-Rips, Čech, alfa y testigo. De estos algoritmos, los primeros tres se encuentran implementados en la paquetería TDA de R y el último en la librería JavaPlex soportada por Matlab. No se incluye el algoritmo Mapper debido a que este es visual y el análisis se lleva a cabo en un servidor sin entorno gráfico.

Nuestro estudio de simulación se elaboró en un servidor dedicado al proyecto de ATD del CIMAT, el cual cuenta con las siguientes especificaciones:

- 24 núcleos (2x12) a 3.4GHz Intel Xeon.
- 128GB de memoria RAM.
- 2.5TB de almacenamiento en disco duro.

Como criterios de comparación, se toman mediciones de las siguientes características:

1. Tiempo de ejecución, medido en segundos utilizados por el CPU.
2. Memoria utilizada por el proceso (física y virtual).

En las tablas que mostramos más adelante se utilizan estos criterios para presentar los resultados de las simulaciones. Éstas se organizan por **tipo** de ruido, tamaño **n** de la muesta utilizada, cantidad de memoria virtual (**VSZ**) y física (**RES**) utilizada por el proceso medidos en GB. Se muestra también el porcentaje de memoria física utilizada (**%**) y el **tiempo** de ejecución en cada caso.

La manera en que añadimos ruido a una muestra es simulando una nube de datos X que viven en $\mathbb{R}^d$ sumando como ruido una variable normal $N(0, \sigma^2)$ en las entradas de cada elemento de la nube de datos, de modo que se sigue el modelo

$$X + \sigma N(0, I_d), \qquad d = \dim(X_i).$$

La notación que utilizaremos en las tablas para el tipo de ruido que se añade a la muestra es la siguiente: NR - sin ruido. 0.005 - $\sigma = 0.005$; 0.05 - $\sigma = 0.05$; 0.1 - $\sigma = 0.1$.

En las tablas que mostramos a continuación se resumen los tiempos de ejecución de cada algoritmo. Existen ocasiones en las que la capacidad de hardware del sistema no fue capaz de soportar el cálculo de la homología, razón por la cual se decidió interrumpirlo. En este caso los recuadros respectivos en cada tabla se dejaron en blanco para indicar que no contamos con recurso computacional suficiente para llevar a cabo dicho cálculo.

5.2.1. $\mathbb{S}^1$ (VR)

Tipo	n	VSZ	RES	%	Tiempo
	500	1.2121	0.1499	0.12	1.5576
	800	1.5049	0.4281	0.34	6.9292
NR	1000	1.8461	0.7694	0.61	14.743
	1500	3.4462	2.3695	1.89	56.36
	2000	6.7511	5.6617	4.51	135.02
	500	1.2030	0.1476	0.12	1.524
	800	1.5106	0.4317	0.34	6.8963
0.005	1000	1.8393	0.7603	0.61	14.21
	1500	2.8520	2.4303	1.93	53.731
	2000	6.7208	5.6420	4.49	132.6
	500	1.2261	0.1474	0.12	1.4418
	800	1.4771	0.3984	0.32	6.3338
0.05	1000	1.8062	0.7275	0.58	13.565
	1500	3.2964	2.2177	1.76	51.412
	2000	6.2702	5.1897	4.13	127.37
	500	1.2083	0.1353	0.11	1.2676
	800	1.4334	0.3604	0.29	5.461
0.1	1000	1.7140	0.6410	0.51	10.86
	1500	3.1075	2.0344	1.62	44.893
	2000	5.7377	4.6624	3.71	108.85

Tabla 5.6: Distribución uniforme sobre $\mathbb{S}^1$. Complejo Vietoris-Rips.

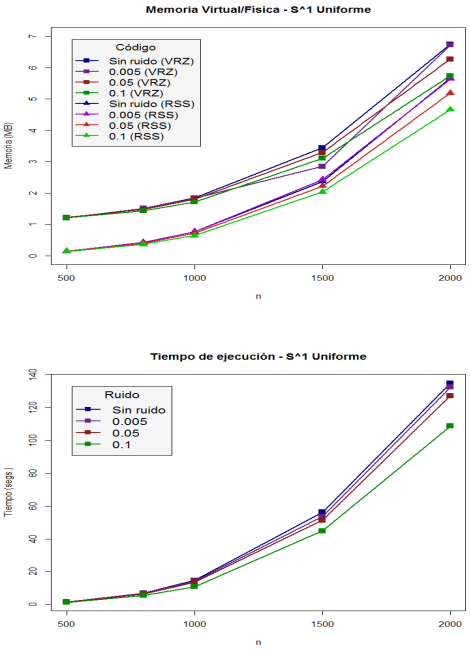


Figura 5.1: Uso de memoria física y virtual.

Tipo	n	VSZ	RES	%	Tiempo
	500	1.1789	0.1426	0.11	1.4878
	800	1.4740	0.4260	0.34	6.7714
NR	1000	1.8074	0.7594	0.60	14.2196
	1500	3.4145	2.3660	1.88	51.6452
	2000	6.7204	5.6694	4.51	129.0258
	500	1.2019	0.1374	0.11	1.5858
	800	1.5007	0.4218	0.34	6.992
0.005	1000	1.8311	0.7520	0.60	14.38
	1500	3.3817	2.2832	1.82	55.0768
	2000	6.5686	5.4875	4.37	131.0394
	500	1.2068	0.1399	0.11	1.4312
	800	1.4887	0.4099	0.33	6.5012
0.05	1000	1.8144	0.7280	0.58	13.7328
	1500	3.3419	2.2644	1.80	48.5942
	2000	6.3218	5.2402	4.17	119.9822
	500	1.2106	0.1339	0.11	1.3312
	800	1.4486	0.3527	0.28	5.7948
0.1	1000	1.7288	0.6520	0.52	10.635
	1500	3.0365	1.9584	1.56	45.9044
	2000	5.8293	4.7509	3.78	112.9446

Tabla 5.7: Distribución concentrada en ejes cartesianos sobre $\mathbb{S}^1$. Complejo Vietoris-Rips.

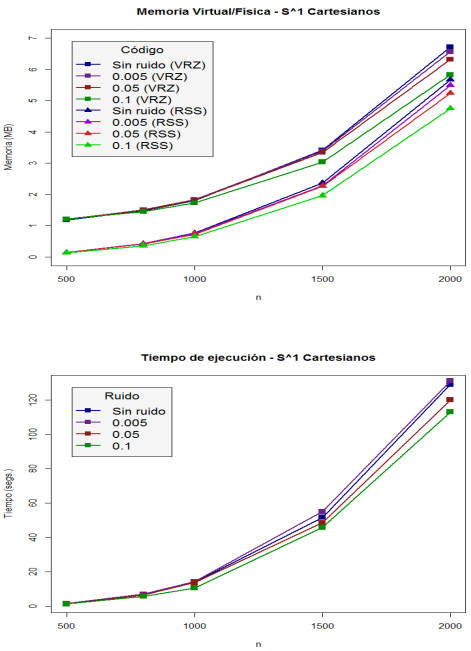


Figura 5.2: Uso de memoria física y virtual.

Tipo	n	VSZ	RES	%	Tiempo
	500	1.2296	0.1503	0.12	1.4366
	800	1.5061	0.4267	0.34	6.4304
NR	1000	1.8145	0.7321	0.58	13.0598
	1500	3.6895	2.6097	2.08	51.9368
	2000	6.7752	5.6933	4.53	136.754
	500	1.2122	0.1468	0.12	1.5354
	800	1.4880	0.4107	0.33	6.9428
0.005	1000	1.8709	0.7937	0.63	13.7836
	1500	3.3074	2.2297	1.77	53.3698
	2000	6.8872	5.8078	4.62	135.514
	500	1.2028	0.1356	0.11	1.2692
	800	1.4603	0.3813	0.30	6.0388
0.05	1000	1.7953	0.7163	0.57	13.0396
	1500	3.4692	2.3898	1.90	50.7554
	2000	6.3060	5.2244	4.16	128.594
	500	1.1949	0.1295	0.10	1.2542
	800	1.4367	0.3597	0.29	5.7306
0.1	1000	1.7046	0.6276	0.50	10.9556
	1500	3.0974	2.0197	1.61	45.91
	2000	5.7247	4.6457	3.70	113.256

Tabla 5.8: Distribución concentrada sobre $\mathbb{S}^1(\rho = 0.1)$. Complejo Vietoris-Rips.

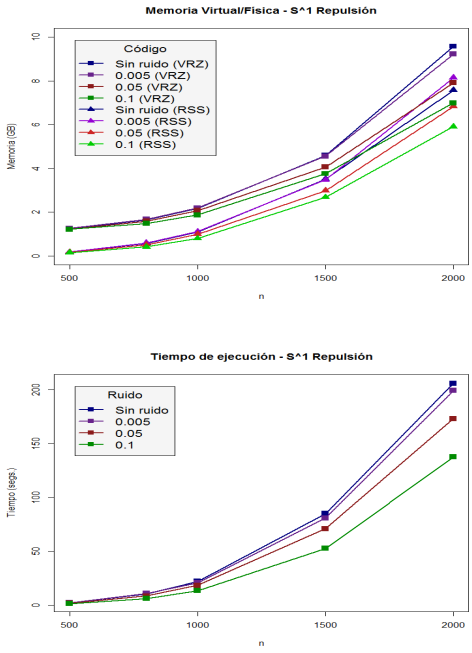


Figura 5.3: Uso de memoria física y virtual.

Tipo	n	VSZ	RES	%	Tiempo
	500	1.3069	0.2278	0.18	2.8092
	800	1.9037	0.8246	0.66	14.4726
NR	1000	2.5973	1.5178	1.21	29.4858
	1500	6.1750	5.0959	4.06	117.8566
	2000	12.5799	11.4973	9.15	303.989
	500	1.2701	0.2244	0.18	2.766
	800	1.8781	0.7989	0.64	15.117
0.005	1000	2.5428	1.4632	1.16	28.7022
	1500	6.0260	4.9469	3.94	114.5736
	2000	12.0132	10.9307	8.70	294.0124
	500	1.2909	0.2116	0.17	2.4502
	800	1.7400	0.6608	0.53	11.9812
0.05	1000	2.3771	1.2977	1.03	25.258
	1500	5.3614	4.2823	3.41	100.169
	2000	11.2372	10.1549	8.08	265.1556
	500	1.2333	0.1834	0.15	1.8596
	800	1.6595	0.5863	0.47	9.386
0.1	1000	2.1456	1.0722	0.85	19.7836
	1500	4.6532	3.5799	2.85	79.8432
	2000	9.0545	7.9353	6.31	204.9982

Tabla 5.9: Distribución concentrada sobre $\mathbb{S}^1$ $(\rho = 0.98)$. Complejo Vietoris-Rips.

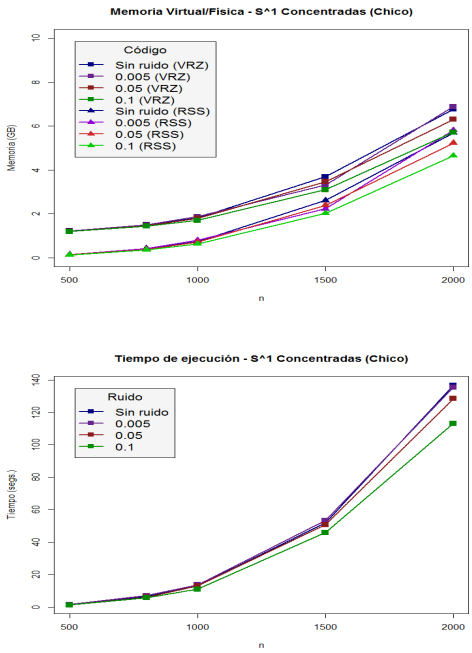


Figura 5.4: Uso de memoria física y virtual.

Tipo	n	VSZ	RES	%	Tiempo
	500	1.2486	0.1820	0.14	2.2444
	800	1.6477	0.5693	0.45	10.4656
NR	1000	2.1778	1.0993	0.87	22.1212
	1500	4.5932	3.5149	2.80	84.9804
	2000	9.5783	7.5779	6.03	205.8644
	500	1.2598	0.1837	0.15	2.3144
	800	1.6714	0.5953	0.47	10.8662
0.005	1000	2.1840	1.1078	0.88	20.744
	1500	4.5535	3.4767	2.77	80.8702
	2000	9.2256	8.1469	6.48	199.1238
	500	1.2211	0.1666	0.13	1.8822
	800	1.5876	0.5097	0.41	8.4411
0.05	1000	2.0743	0.9961	0.79	18.3672
	1500	4.0610	2.9831	2.37	70.8376
	2000	7.9107	6.8303	5.44	173.0084
	500	1.2142	0.1511	0.12	1.4298
	800	1.4838	0.4089	0.33	6.1982
0.1	1000	1.8715	0.7926	0.63	13.4892
	1500	3.7613	2.6826	2.13	52.6162
	2000	6.9911	5.9101	4.70	137.6002

Tabla 5.10: Distribución con repulsión sobre $\mathbb{S}^1$. Complejo Vietoris-Rips.

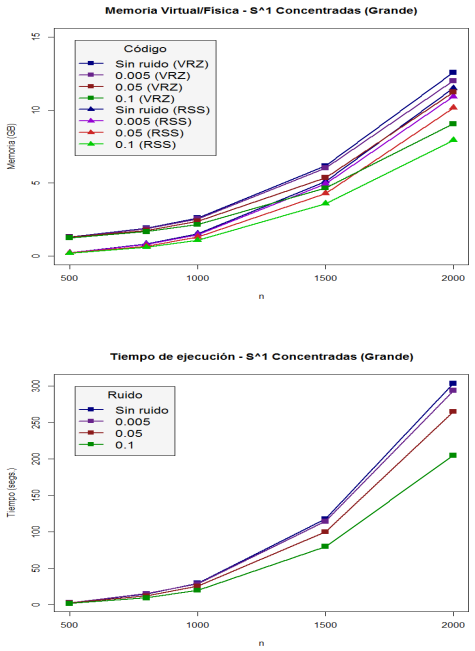
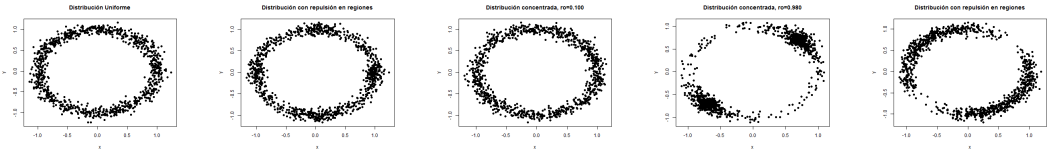


Figura 5.5: Uso de memoria física y virtual.



(a) $\mathbb{S}^1$ Uniforme (b) $\mathbb{S}^1$ Cartesianos (c) $\mathbb{S}^1$ Regiones (d) $\mathbb{S}^1$ Regiones (e) $\mathbb{S}^1$ Repulsión

Figura 5.6: Distribuciones sobre $\mathbb{S}^1$, ruido añadido $\sigma = 0.005$.

En las Tablas 5.6–5.10 podemos observar que para cada tipo de ruido aumenta el tamaño de muestra se incrementa el uso de memoria así como el tiempo de ejecución. Hecho que era de esperarse. Por otro lado, cuando se incrementa el ruido sobre cada distribución, el uso de recursos y el tiempo de ejecución tiende a disminuir. Sin embargo, esto último puede presentar un problema en muestras reales pues si existe mucho ruido en la misma, será difícil para los algoritmos capturar de manera correcta las características geométricas y topológicas.

En las Tablas 5.6–5.8 es posible observar que el rendimiento computacional es parecido, pues como se observa en la Figura 5.6 la distribución en estos tres casos tiene un comportamiento "similar". El costo y tiempo de ejecución aumentan cuando tenemos distribuciones con mayor concentración en algunas regiones y dispersión de puntos en otras (Tablas 5.9 y 5.10); debido a esta concentración, se crean grupos con mayor número de simplejos a diferencia de las distribuciones con mayor parecido a la uniforme.

5.2.2. $\mathbb{S}^1$(MS)

Tipo	n	VSZ	RES	%	Tiempo
	500	1.1340	0.0588	0.05	0.5018
	800	1.1366	0.0597	0.05	0.7746
NR	1000	1.1373	0.0636	0.05	0.957
	1500	1.1368	0.0579	0.05	1.4068
	2000	1.1323	0.0591	0.05	1.8544
	500	1.1340	0.0589	0.05	0.4982
	800	1.1403	0.0578	0.05	0.7726
0.005	1000	1.1374	0.0598	0.05	0.9554
	1500	1.1349	0.0601	0.05	1.4522
	2000	1.1367	0.0587	0.05	1.8554
	500	1.1336	0.0623	0.05	0.4992
	800	1.1362	0.0594	0.05	0.7718
0.05	1000	1.1356	0.0577	0.05	0.9562
	1500	1.1350	0.0620	0.05	1.4072
	2000	1.1350	0.0602	0.05	1.8592
	500	1.1341	0.0569	0.05	0.497
	800	1.1358	0.0570	0.05	0.7728
0.1	1000	1.1376	0.0587	0.05	0.9802
	1500	1.1331	0.0585	0.05	1.4102
	2000	1.1343	0.0650	0.05	1.8598

Tabla 5.11: Distribución uniforme sobre $\mathbb{S}^1$. Complejo Morse-Smale.

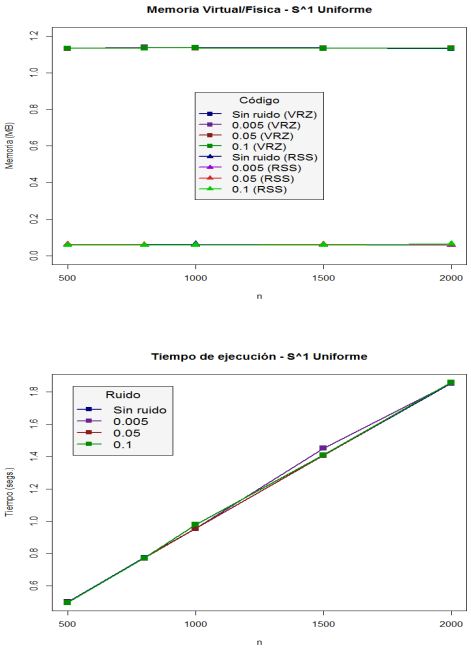


Figura 5.7: Uso de memoria física y virtual.

Tipo	n	VSZ	RES	%	Tiempo
	500	1.1026	0.0562	0.04	0.499
	800	1.1048	0.0565	0.04	0.775
NR	1000	1.1058	0.0571	0.05	0.9808
	1500	1.1041	0.0598	0.05	1.4016
	2000	1.1041	0.0591	0.05	1.857
	500	1.1142	0.0570	0.05	0.4998
	800	1.1273	0.0576	0.05	0.779
0.005	1000	1.1371	0.0576	0.05	0.9542
	1500	1.1342	0.0595	0.05	1.4076
	2000	1.1362	0.0616	0.05	1.8576
	500	1.1332	0.0606	0.05	0.4996
	800	1.1360	0.0597	0.05	0.7744
0.05	1000	1.1371	0.0595	0.05	0.9554
	1500	1.1336	0.0590	0.05	1.403
	2000	1.1375	0.0596	0.05	1.8612
	500	1.1400	0.0574	0.05	0.4986
	800	1.1360	0.0593	0.05	0.7814
0.1	1000	1.1315	0.0567	0.05	0.9558
	1500	1.1332	0.0589	0.05	1.4078
	2000	1.1374	0.0616	0.05	1.858

Tabla 5.12: Distribución concentrada en ejes cartesianos sobre $\mathbb{S}^1$. Complejo Morse-Smale.

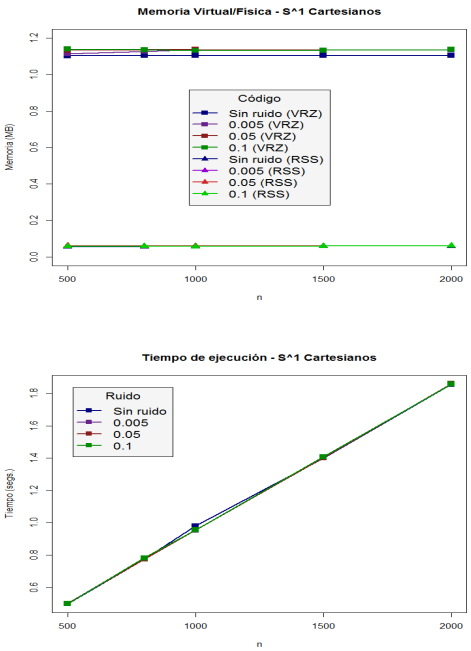


Figura 5.8: Uso de memoria física y virtual.

Tipo	n	VIRT	RES	%	Tiempo
	500	1.1526	0.0698	0.06	0.5064
	800	1.1525	0.0674	0.05	0.7808
NR	1000	1.1532	0.0704	0.06	0.96
	1500	1.1540	0.0692	0.06	1.4136
	2000	1.1476	0.0683	0.05	1.8724
	500	1.1527	0.0678	0.05	0.5106
	800	1.1529	0.0678	0.05	0.7818
0.005	1000	1.1531	0.0683	0.05	0.9654
	1500	1.1469	0.0675	0.05	1.4608
	2000	1.1554	0.0704	0.06	1.8938
	500	1.1532	0.0722	0.06	0.5084
	800	1.1526	0.0698	0.06	0.7778
0.05	1000	1.1529	0.0698	0.06	0.9614
	1500	1.1469	0.0698	0.06	1.4162
	2000	1.1487	0.0719	0.06	1.8664
	500	1.1526	0.0677	0.05	0.5126
	800	1.1530	0.0694	0.06	0.7788
0.1	1000	1.1533	0.0681	0.05	0.962
	1500	1.1474	0.0701	0.06	1.4216
	2000	1.1476	0.0685	0.05	1.872

Tabla 5.13: Distribución concentrada en regiones sobre $\mathbb{S}^1$ ($\rho = 0.1$). Complejo Morse-Smale.

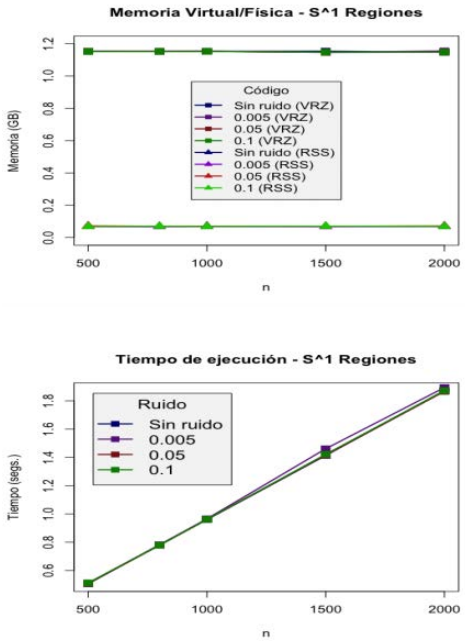


Figura 5.9: Uso de memoria física y virtual.

Tipo	n	VIRT	RES	%	Tiempo
	500	1.1524	0.0698	0.06	0.5072
	800	1.1520	0.0711	0.06	0.7822
NR	1000	1.1527	0.0681	0.05	0.9662
	1500	1.1475	0.0701	0.06	1.4168
	2000	1.1476	0.0683	0.05	1.874
	500	1.1529	0.0679	0.05	0.5102
	800	1.1527	0.0681	0.05	0.7944
0.005	1000	1.1528	0.0697	0.06	0.9638
	1500	1.1469	0.0677	0.05	1.417
	2000	1.1488	0.0720	0.06	1.8738
	500	1.1532	0.0681	0.05	0.5086
	800	1.1525	0.0679	0.05	0.7814
0.05	1000	1.1530	0.0699	0.06	0.9616
	1500	1.1469	0.0696	0.06	1.4152
	2000	1.1486	0.0714	0.06	1.8788
	500	1.1526	0.0695	0.06	0.5076
	800	1.1527	0.0676	0.05	0.7814
0.1	1000	1.1529	0.0701	0.06	0.9604
	1500	1.1469	0.0696	0.06	1.4258
	2000	1.1487	0.0733	0.06	1.8828

Tabla 5.14: Distribución concentrada en regiones sobre $\mathbb{S}^1$ ($\rho = 0.98$). Complejo Morse-Smale.

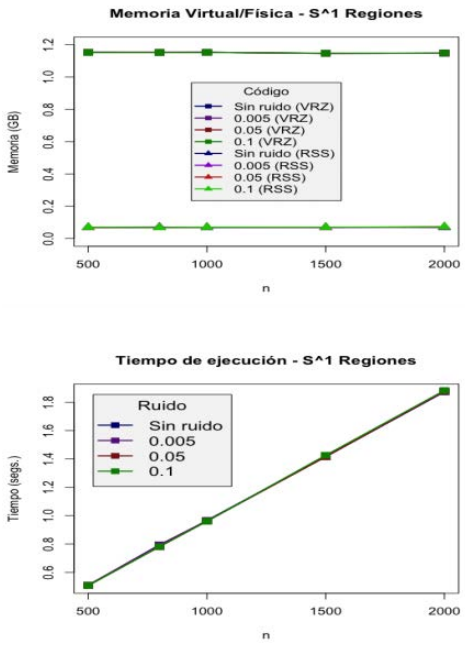


Figura 5.10: Uso de memoria física y virtual.

Tipo	n	VIRT	RES	%	Tiempo
	500	1.1605	0.0759	0.06	0.5236
	800	1.1603	0.0755	0.06	0.798
NR	1000	1.1535	0.0745	0.06	0.9786
	1500	1.1559	0.0772	0.06	1.432
	2000	1.1490	0.0761	0.06	1.8866
	500	1.1611	0.0767	0.06	0.5248
	800	1.1602	0.0778	0.06	0.7956
0.005	1000	1.1607	0.0785	0.06	0.9812
	1500	1.1627	0.0784	0.06	1.4318
	2000	1.1498	0.0790	0.06	1.8824
	500	1.1611	0.0806	0.06	0.5284
	800	1.1537	0.0791	0.06	0.799
0.05	1000	1.1538	0.0757	0.06	0.9752
	1500	1.1555	0.0793	0.06	1.4356
	2000	1.1497	0.0790	0.06	1.8874
	500	1.1609	0.0768	0.06	0.5194
	800	1.1604	0.0819	0.07	0.8
0.1	1000	1.1536	0.0753	0.06	0.9802
	1500	1.1555	0.0787	0.06	1.4324
	2000	1.1498	0.0789	0.06	1.886

Tabla 5.15: Distribución con repulsión sobre $\mathbb{S}^1$. Complejo Morse-Smale.

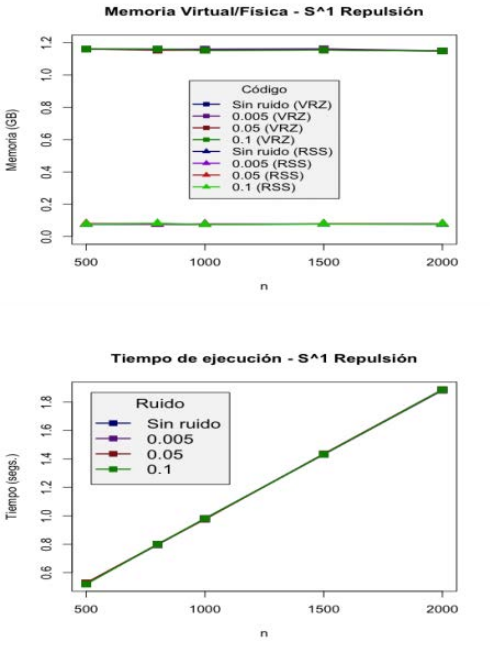


Figura 5.11: Uso de memoria física y virtual.

Como podemos observar en las tablas 5.11-5.15. El uso de memoria se mantiene constante sin importar el tamaño de ruido, el tamaño de la muestra o la distribución de los datos. Este uso de los recursos puede deberse al tamaño "pequeño" de la filtración (comparado con el algoritmo VR).

Por otro lado, es posible observar también que el tiempo de ejecución tiene un comportamiento lineal con respecto al tamaño de muestra. Esto, junto con el uso de recursos y la detección de la geometría es un asunto relevante a tener en cuenta al llevar a cabo análisis de la topología utilizando este algoritmo.

5.2.3. $\mathbb{S}^1_{\text{(alfa)}}$

Tipo	n	VIRT	RES	%	Tiempo
	500	1.0811	0.0595	0.05	0.08208
	800	1.0815	0.0558	0.04	0.12682
NR	1000	1.0815	0.0561	0.04	0.15692
	1500	1.0823	0.0570	0.05	0.2294
	2000	1.0823	0.0566	0.05	0.31178
	500	1.0819	0.0563	0.04	0.13896
	800	1.0818	0.0581	0.05	0.22404
0.005	1000	1.0821	0.0604	0.05	0.28086
	1500	1.0835	0.0600	0.05	0.42762
	2000	1.0832	0.0598	0.05	0.58868
	500	1.0821	0.0621	0.05	0.14356
	800	1.0820	0.0585	0.05	0.22716
0.05	1000	1.0821	0.0587	0.05	0.2851
	1500	1.0836	0.0581	0.05	0.44178
	2000	1.0833	0.0580	0.05	0.58356
	500	1.0821	0.0566	0.05	0.14088
	800	1.0820	0.0563	0.04	0.22724
0.1	1000	1.0821	0.0584	0.05	0.28512
	1500	1.0836	0.0599	0.05	0.4337
	2000	1.0835	0.0599	0.05	0.59276

Tabla 5.16: Distribución uniforme sobre $\mathbb{S}^1$. Complejo alfa.

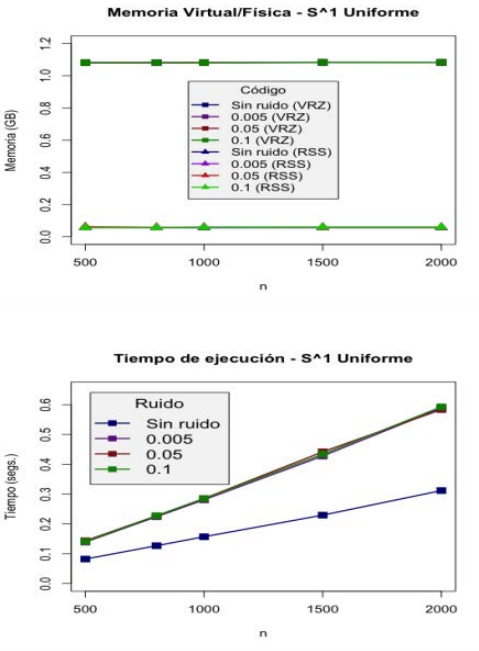


Figura 5.12: Uso de memoria física y virtual.

Tipo	n	VIRT	RES	%	Tiempo
	500	1.0527	0.0580	0.05	0.08244
	800	1.0528	0.0598	0.05	0.12568
NR	1000	1.0530	0.0578	0.05	0.2324
	1500	1.0530	0.0583	0.05	0.2565
	2000	1.0538	0.0609	0.05	0.30036
	500	1.0839	0.0620	0.05	0.13856
	800	1.0839	0.0605	0.05	0.22444
0.005	1000	1.0840	0.0586	0.05	0.29406
	1500	1.0845	0.0628	0.05	0.4281
	2000	1.0846	0.0610	0.05	0.57062
	500	1.0838	0.0581	0.05	0.14052
	800	1.0840	0.0607	0.05	0.22764
0.05	1000	1.0841	0.0586	0.05	0.28276
	1500	1.0846	0.0609	0.05	0.42118
	2000	1.0845	0.0609	0.05	0.57302
	500	1.0837	0.0600	0.05	0.14168
	800	1.0838	0.0601	0.05	0.2278
0.1	1000	1.0839	0.0602	0.05	0.28958
	1500	1.0848	0.0615	0.05	0.42924
	2000	1.0845	0.0609	0.05	0.57252

Tabla 5.17: Distribución concentrada en ejes cartesianos sobre $\mathbb{S}^1$. Complejo alfa.

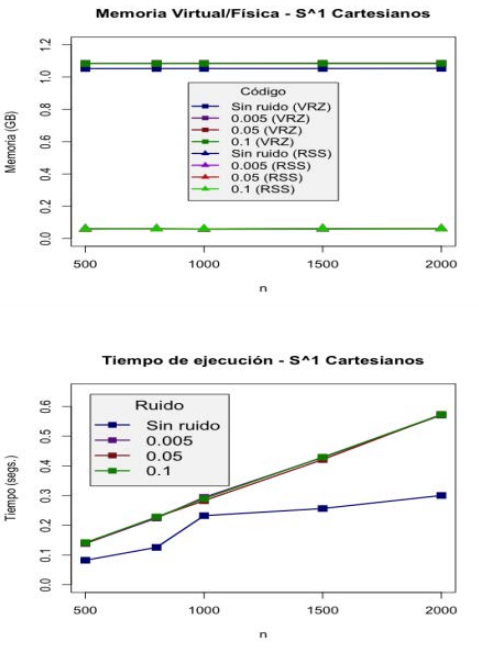


Figura 5.13: Uso de memoria física y virtual.

Tipo	n	VIRT	RES	%	Tiempo
	500	1.0812	0.0555	0.04	0.08262
	800	1.0816	0.0560	0.04	0.12692
NR	1000	1.0814	0.0596	0.05	0.15716
	1500	1.0824	0.0587	0.05	0.2297
	2000	1.0830	0.0592	0.05	0.30442
	500	1.0821	0.0583	0.05	0.13916
	800	1.0821	0.0565	0.04	0.2244
0.005	1000	1.0821	0.0587	0.05	0.28536
	1500	1.0837	0.0599	0.05	0.42758
	2000	1.0833	0.0634	0.05	0.57596
	500	1.0822	0.0565	0.04	0.1414
	800	1.0821	0.0563	0.04	0.22878
0.05	1000	1.0822	0.0569	0.05	0.28858
	1500	1.0834	0.0578	0.05	0.43226
	2000	1.0834	0.0577	0.05	0.58768
	500	1.0822	0.0583	0.05	0.14124
	800	1.0822	0.0606	0.05	0.2262
0.1	1000	1.0822	0.0569	0.05	0.2859
	1500	1.0835	0.0601	0.05	0.43076
	2000	1.0833	0.0580	0.05	0.5781

Tabla 5.18: Distribución concentrada en regiones sobre $\mathbb{S}^1$ ($\rho = 0.1$). Complejo alfa.

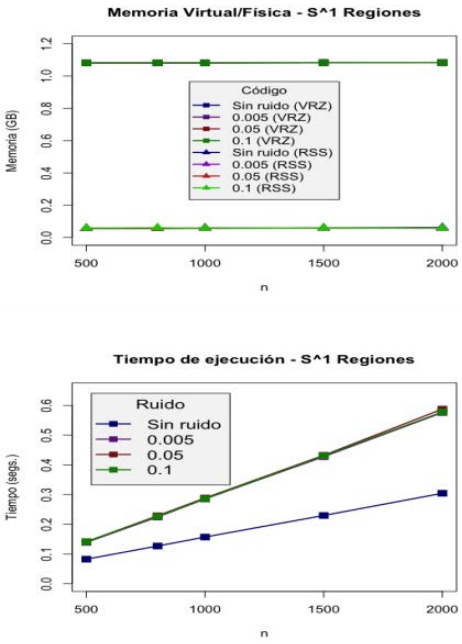


Figura 5.14: Uso de memoria física y virtual.

Tipo	n	VIRT	RES	%	Tiempo
	500	1.0812	0.0555	0.04	0.07946
	800	1.0814	0.0555	0.04	0.12372
NR	1000	1.0813	0.0557	0.04	0.15196
	1500	1.0830	0.0592	0.05	0.22464
	2000	1.0833	0.0595	0.05	0.29912
	500	1.0815	0.0577	0.05	0.1391
	800	1.0816	0.0559	0.04	0.23044
0.005	1000	1.0822	0.0584	0.05	0.2856
	1500	1.0834	0.0580	0.05	0.43022
	2000	1.0832	0.0595	0.05	0.59438
	500	1.0816	0.0578	0.05	0.14256
	800	1.0815	0.0596	0.05	0.22632
0.05	1000	1.0822	0.0584	0.05	0.28582
	1500	1.0834	0.0599	0.05	0.43458
	2000	1.0832	0.0597	0.05	0.5882
	500	1.0817	0.0637	0.05	0.14094
	800	1.0815	0.0619	0.05	0.22972
0.1	1000	1.0828	0.0570	0.05	0.28424
	1500	1.0838	0.0581	0.05	0.4368
	2000	1.0832	0.0595	0.05	0.58318

Tabla 5.19: Distribución concentrada en regiones sobre $\mathbb{S}^1$ ($\rho = 0.98$). Complejo alfa.

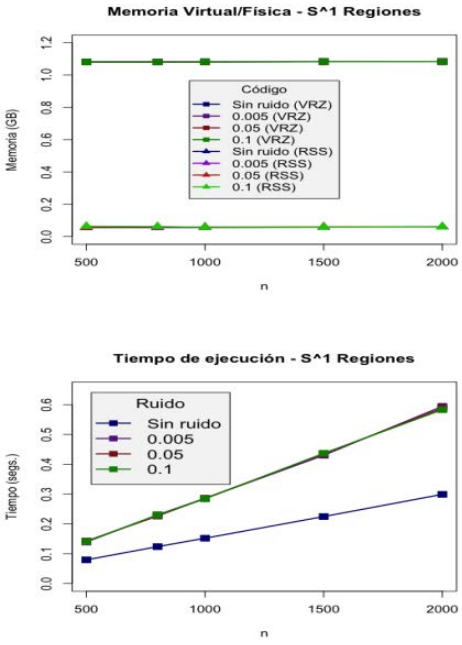


Figura 5.15: Uso de memoria física y virtual.

Tipo	n	VIRT	RES	%	Tiempo
	500	1.0887	0.0632	0.05	0.08398
	800	1.0890	0.0637	0.05	0.12748
NR	1000	1.0891	0.0659	0.05	0.15742
	1500	1.0893	0.0642	0.05	0.23246
	2000	1.0896	0.0682	0.05	0.305
	500	1.0889	0.0659	0.05	0.14218
	800	1.0894	0.0663	0.05	0.22996
0.005	1000	1.0895	0.0667	0.05	0.28306
	1500	1.0901	0.0672	0.05	0.43292
	2000	1.0905	0.0656	0.05	0.58034
	500	1.0889	0.0659	0.05	0.14334
	800	1.0893	0.0664	0.05	0.23094
0.05	1000	1.0885	0.0634	0.05	0.28992
	1500	1.0901	0.0650	0.05	0.43294
	2000	1.0905	0.0676	0.05	0.58798
	500	1.0889	0.0639	0.05	0.14476
	800	1.0894	0.0723	0.06	0.23238
0.1	1000	1.0894	0.0645	0.05	0.28972
	1500	1.0902	0.0653	0.05	0.4364
	2000	1.0905	0.0675	0.05	0.57896

Tabla 5.20: Distribución con repulsión sobre $\mathbb{S}^1$. Complejo alfa.

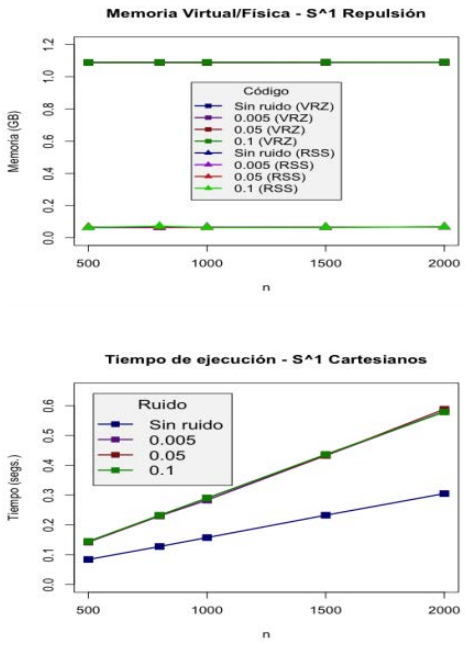


Figura 5.16: Uso de memoria física y virtual.

En las tablas 5.16-5.20 podemos ver que el comportamiento del algoritmo alfa es similar al de los complejos de Morse-Smale en cuanto al uso de recursos. No importa el tamaño de muestra, el ruido añadido o la distribución de los datos.

Asimismo, el tiempo de ejecución del algoritmo tiene un comportamiento lineal con respecto al tamaño de muestra, pero se ve reducido más alla de la mitad en comparación con el tiempo de ejecución del algoritmo MS.

5.2.4. $\mathbb{S}^2$

Tipo	n	VSZ	RES	%	Tiempo
	500	1.3393	0.2677	0.21	3.072
	800	2.3011	1.2163	0.97	22.9962
NR	1000	3.8130	2.7416	2.18	58.8928
	1500	14.5967	13.5314	10.77	334.2844
	2000	42.3470	41.0084	32.63	1165.718
	500	1.3310	0.2576	0.20	2.7288
	800	2.3071	1.2426	0.99	22.3574
0.005	1000	3.8802	2.8069	2.23	56.6408
	1500	14.2826	13.2093	10.51	322.2084
	2000	42.1841	40.6865	32.38	1181.5052
	500	1.2470	0.1775	0.14	1.9654
	800	1.8490	0.7756	0.62	14.9138
0.05	1000	2.8371	1.7638	1.40	36.3692
	1500	9.1168	8.0534	6.41	199.6148
	2000	27.0171	25.9458	20.65	740.1476
	500	1.2053	0.1339	0.11	1.5846
	800	1.6050	0.5318	0.42	12.5724
0.1	1000	2.2249	1.1536	0.92	31.0104
	1500	6.4162	5.3330	4.24	120.4958
	2000	16.1795	15.1532	12.06	410.2976

Tabla 5.21: Distribución uniforme sobre $\mathbb{S}^2$. Complejo Vietoris-Rips.

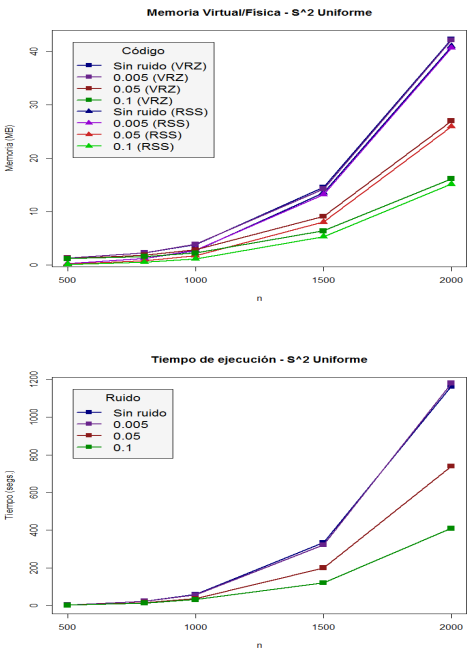


Figura 5.17: Uso de memoria física y virtual.

Tipo	n	VSZ	RES	%	Tiempo
	500	1.3224	0.3267	0.26	4.5036
	800	3.0016	2.0061	1.60	37.7988
NR	1000	5.6221	4.6283	3.68	98.028
	1500	22.3116	21.3160	16.96	560.482
	2000	65.5618	63.3985	50.45	2068.5846
	500	1.3447	0.3226	0.26	4.5868
	800	2.7015	1.6772	1.33	34.646
0.005	1000	4.9452	3.9210	3.12	89.1974
	1500	21.4575	20.4333	16.26	523.707
	2000	63.4005	62.3762	49.64	1879.0136
	500	1.2207	0.1963	0.16	2.4014
	800	1.9411	0.9187	0.73	18.3542
0.05	1000	3.2716	2.2474	1.79	51.642
	1500	10.9130	9.8867	7.87	363.2566
	2000	34.2186	33.1945	26.42	901.8432
	500	1.1661	0.1417	0.11	1.2158
	800	1.5829	0.5587	0.44	9.1316
0.1	1000	2.2182	1.1921	0.95	22.65
	1500	6.8206	5.7944	4.61	128.5916
	2000	18.9636	17.9373	14.27	465.8152

Tabla 5.22: Distribución concentrada en ejes cartesianos sobre $\mathbb{S}^2$. Complejo Vietoris-Rips.

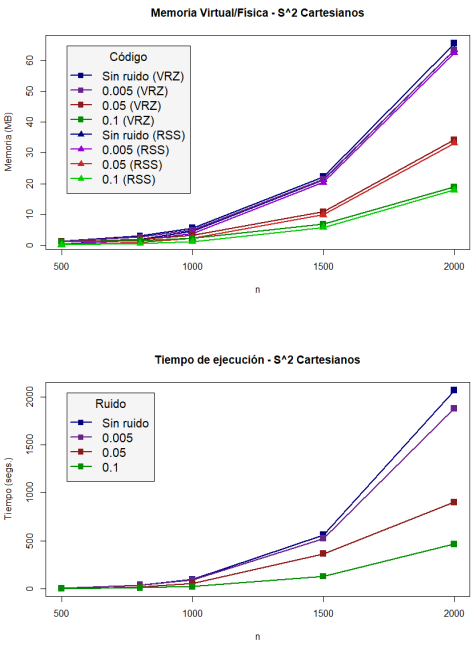


Figura 5.18: Uso de memoria física y virtual.

5.2.5. $\mathbb{S}^2{}_{(\mathrm{MS})}$

Tipo	n	VSZ	RES	%	Tiempo
	500	1.2944	0.2452	0.20	3.512
	800	1.2907	0.2399	0.19	3.5336
NR	1000	1.2821	0.2547	0.20	3.5472
	1500	1.2905	0.2446	0.19	3.6712
	2000	1.2761	0.2253	0.18	3.7048
	500	1.2853	0.2356	0.19	3.4688
	800	1.2906	0.2416	0.19	3.5304
0.005	1000	1.2800	0.2297	0.18	3.5776
	1500	1.2843	0.2325	0.19	3.596
	2000	1.2737	0.2266	0.18	3.6176
	500	1.2947	0.2471	0.20	3.36
	800	1.2886	0.2807	0.22	3.3672
0.05	1000	1.2953	0.2453	0.20	3.3736
	1500	1.3000	0.2373	0.19	3.4032
	2000	1.2752	0.2268	0.18	3.6264
	500	1.3089	0.2632	0.21	3.152
	800	1.3016	0.2495	0.20	3.1584
0.1	1000	1.2967	0.2508	0.20	3.1896
	1500	1.3630	0.2806	0.22	3.3272
	2000	1.2807	0.2296	0.18	3.3608

Tabla 5.23: Distribución uniforme sobre $\mathbb{S}^2$. Complejo Morse-Smale.

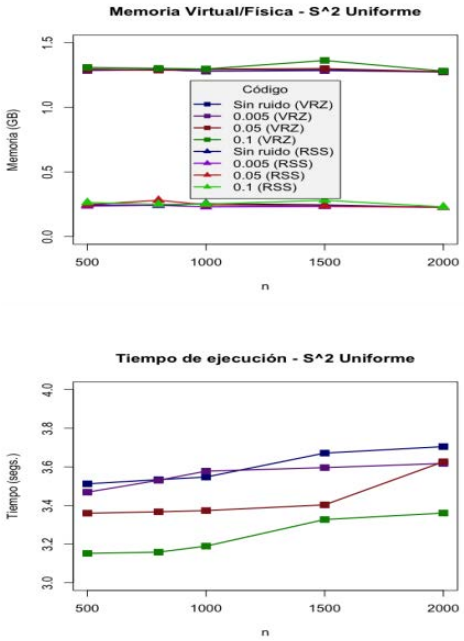


Figura 5.19: Uso de memoria física y virtual.

Tipo	n	VSZ	RES	%	Tiempo
	500	1.2629	0.2427	0.19	3.3776
	800	1.2761	0.2438	0.19	3.3696
NR	1000	1.2793	0.2463	0.20	3.4056
	1500	1.2724	0.2531	0.20	3.4256
	2000	1.2754	0.2559	0.20	3.5448
	500	1.3169	0.2662	0.21	3.3144
	800	1.2977	0.2458	0.20	3.3456
0.005	1000	1.2951	0.2432	0.19	3.3904
	1500	1.2831	0.2222	0.18	3.392
	2000	1.2940	0.2431	0.19	3.4616
	500	1.2984	0.2363	0.19	3.2112
	800	1.3110	0.2599	0.21	3.1776
0.05	1000	1.2925	0.2298	0.18	3.2728
	1500	1.2991	0.2493	0.20	3.2744
	2000	1.2860	0.2273	0.18	3.3008
	500	1.3110	0.2608	0.21	3.128
	800	1.3106	0.2593	0.21	3.1472
0.1	1000	1.3098	0.2590	0.21	3.1425
	1500	1.3067	0.2557	0.20	3.2104
	2000	1.2853	0.2245	0.18	3.2216

Tabla 5.24: Distribución concentrada en ejes cartesianos sobre $\mathbb{S}^2$. Complejo MS.

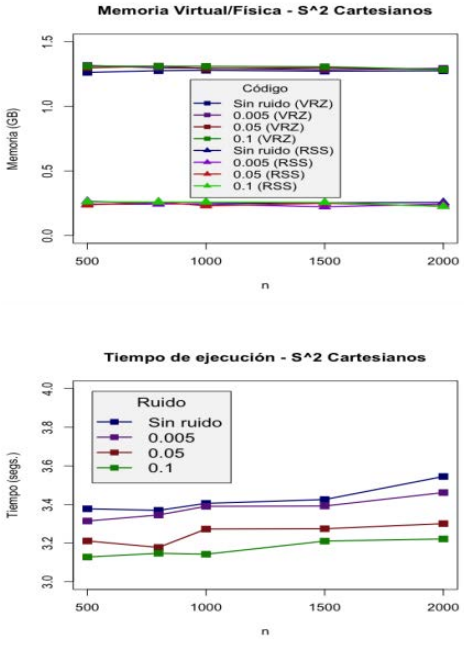


Figura 5.20: Uso de memoria física y virtual.

5.2.6. $\mathbb{S}^2_{(\text{alfa})}$

Tipo	n	VSZ	RES	%	Tiempo
	500	1.0803	0.0566	0.05	1.47022
	800	1.0807	0.0590	0.05	2.37558
NR	1000	1.0811	0.0555	0.04	2.97224
	1500	1.0824	0.0572	0.05	4.46262
	2000	1.0832	0.0575	0.05	5.89762
	500	1.0814	0.0558	0.04	2.24298
	800	1.0813	0.0577	0.05	3.75406
0.005	1000	1.0817	0.0560	0.04	4.79064
	1500	1.0837	0.0599	0.05	7.35202
	2000	1.0849	0.0631	0.05	9.9177
	500	1.0804	0.0550	0.04	2.51374
	800	1.0810	0.0593	0.05	4.19622
0.05	1000	1.0814	0.0558	0.04	5.46266
	1500	1.0839	0.0604	0.05	8.19436
	2000	1.0848	0.0592	0.05	11.05874
	500	1.0802	0.0565	0.04	2.6524
	800	1.0808	0.0592	0.05	4.36754
0.1	1000	1.0813	0.0616	0.05	5.46624
	1500	1.0839	0.0603	0.05	8.34578
	2000	1.0849	0.0613	0.05	11.0644

Tabla 5.25: Distribución uniforme sobre $\mathbb{S}^2$. Complejo alfa.

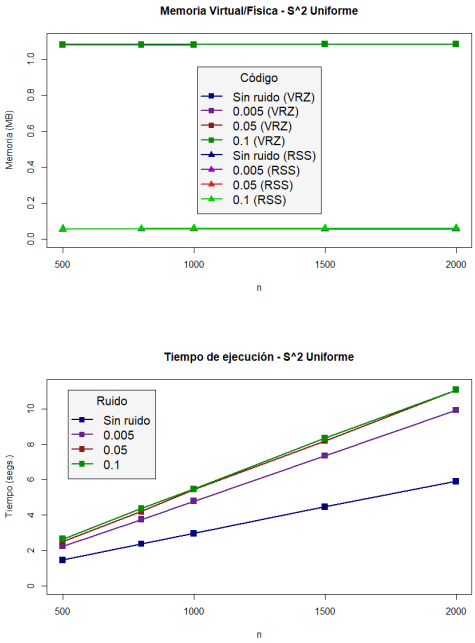


Figura 5.21: Uso de memoria física y virtual.

Tipo	n	VSZ	RES	%	Tiempo
	500	1.0532	0.0604	0.05	1.45476
	800	1.0516	0.0569	0.05	2.3153
NR	1000	1.0514	0.0567	0.05	2.88634
	1500	1.0517	0.0573	0.05	4.2971
	2000	1.0519	0.0573	0.05	5.74112
	500	1.0827	0.0572	0.05	2.23072
	800	1.0835	0.0604	0.05	3.77884
0.005	1000	1.0863	0.0614	0.05	4.86034
	1500	1.0838	0.0603	0.05	7.59048
	2000	1.0850	0.0615	0.05	10.68334
	500	1.0842	0.0588	0.05	2.53026
	800	1.0828	0.0574	0.05	4.30834
0.05	1000	1.0843	0.0588	0.05	5.60482
	1500	1.0839	0.0585	0.05	8.67308
	2000	1.0847	0.0597	0.05	11.58434
	500	1.0832	0.0597	0.05	2.8915
	800	1.0828	0.0573	0.05	4.74522
0.1	1000	1.0842	0.0649	0.05	5.86626
	1500	1.0838	0.0602	0.05	8.59676
	2000	1.0849	0.0638	0.05	11.27706

Tabla 5.26: Distribución concentrada en ejes cartesianos sobre $\mathbb{S}^2$. Complejo alfa.

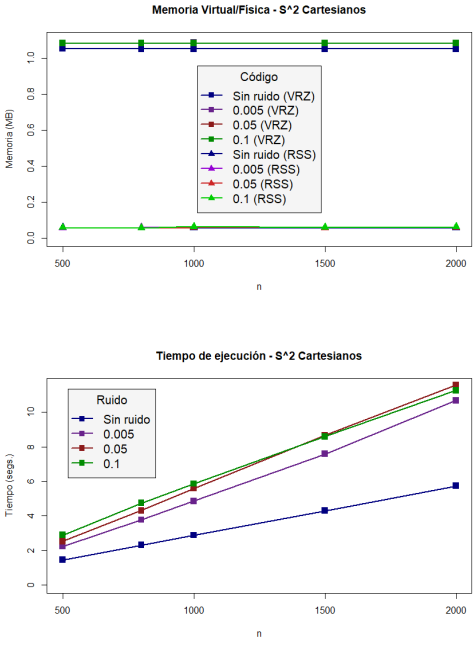


Figura 5.22: Uso de memoria física y virtual.

5.2.7. $\mathbb{T}^2$

Tipo	n	VSZ	RES	%	Tiempo
NR	500	1.1661	0.1396	0.11	1.2808
	800	1.6474	0.6209	0.49	9.9062
	1000	2.3432	1.3205	1.05	25.4358
	1500	7.1599	6.1352	4.88	147.2594
	2000	21.5617	20.5372	16.34	515.7376
0.005	500	1.1688	0.1441	0.11	1.3662
	800	1.6266	0.6022	0.48	10.1368
	1000	2.3893	1.3628	1.08	27.2188
	1500	7.0297	6.0032	4.78	148.0682
	2000	19.2975	18.2711	14.54	482.0208
0.05	500	1.1420	0.1175	0.09	0.9194
	800	1.4897	0.4650	0.37	7.1884
	1000	1.9414	0.9171	0.73	17.7424
	1500	5.5760	4.5615	3.63	99.7906
	2000	14.2498	13.2254	10.52	331.055
0.1	500	1.1188	0.0962	0.08	0.5276
	800	1.3350	0.3105	0.25	4.531
	1000	1.6299	0.5996	0.48	11.4604
	1500	3.9433	2.9190	2.32	62.7228
	2000	9.8199	8.7955	7.00	214.1866

Tabla 5.27: Distribución uniforme sobre $\mathbb{T}^2$. Complejo Vietoris-Rips.

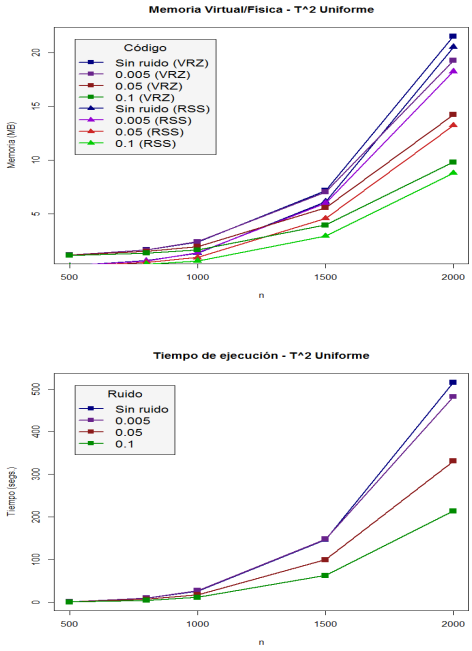


Figura 5.23: Uso de memoria física y virtual.

Tipo	n	VSZ	RES	%	Tiempo
NR	500	1.2587	0.2362	0.19	2.4472
	800	2.0458	1.0234	0.81	16.4082
	1000	3.3194	2.2934	1.83	40.7268
	1500	11.6536	10.2868	8.19	304.8722
	2000	27.7220	26.6917	21.24	929.057
0.005	500	1.2173	0.1915	0.15	2.2418
	800	2.0270	1.0012	0.80	17.736
	1000	3.1150	2.0889	1.66	41.612
	1500	10.8714	9.8476	7.84	282.368
	2000	26.3962	25.3723	20.19	848.609
0.05	500	1.2328	0.2070	0.16	2.069
	800	1.8970	0.8712	0.69	16.553
	1000	3.0671	2.0413	1.62	49.556
	1500	11.7594	10.7352	8.54	279.055
	2000	26.6181	25.5921	20.37	768.109
0.1	500	1.2317	0.2095	0.17	2.2094
	800	1.8069	0.7827	0.62	19.5338
	1000	2.7531	1.7029	1.36	44.4634
	1500	9.2937	8.2695	6.58	250.059
	2000	26.9026	25.8764	20.59	719.52

Tabla 5.28: Distribución concentrada en ejes cartesianos sobre $\mathbb{T}^2$. Complejo Vietoris-Rips.

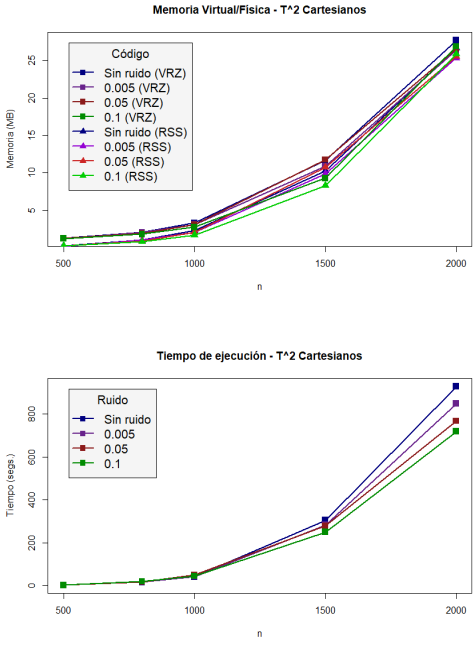


Figura 5.24: Uso de memoria física y virtual.

5.2.8. $\mathbb{T}^2_{(alfa)}$

Tipo	n	VSZ	RES	%	Tiempo
	500	1.0815	0.0558	0.04	3.19668
	800	1.0826	0.0592	0.05	5.54416
NR	1000	1.0840	0.0587	0.05	7.18926
	1500	1.0864	0.0612	0.05	12.47462
	2000	1.0891	0.0636	0.05	17.16668
	500	1.0813	0.0597	0.05	2.31148
	800	1.0820	0.0566	0.05	3.76498
0.005	1000	1.0828	0.0573	0.05	4.81788
	1500	1.0845	0.0592	0.05	8.31082
	2000	1.0860	0.0624	0.05	10.9718
	500	1.0813	0.0599	0.05	2.6344
	800	1.0818	0.0602	0.05	4.6509
0.05	1000	1.0829	0.0594	0.05	6.26004
	1500	1.0843	0.0610	0.05	9.38816
	2000	1.0857	0.0621	0.05	12.6613
	500	1.0813	0.0598	0.05	3.28346
	800	1.0818	0.0564	0.04	5.29612
0.1	1000	1.0829	0.0595	0.05	6.38652
	1500	1.0841	0.0604	0.05	9.63164
	2000	1.0858	0.0625	0.05	13.42804

Tabla 5.29: Distribución uniforme sobre $\mathbb{T}^2$. Complejo alfa.

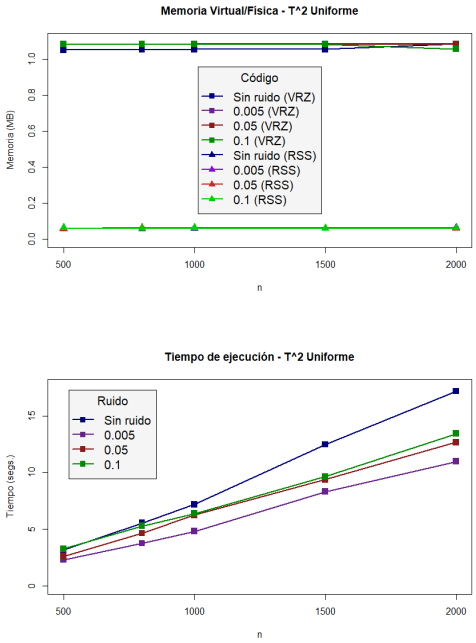


Figura 5.25: Uso de memoria física y virtual.

Tipo	n	VSZ	RES	%	Tiempo
	500	1.0514	0.0588	0.05	3.3493
	800	1.0533	0.0608	0.05	6.77838
NR	1000	1.0549	0.0600	0.05	8.9464
	1500	1.0548	0.0601	0.05	14.6721
	2000	1.0852	0.0637	0.05	18.66806
	500	1.0833	0.0621	0.05	2.69764
	800	1.0837	0.0581	0.05	4.77144
0.005	1000	1.0849	0.0633	0.05	6.2199
	1500	1.0842	0.0612	0.05	9.50792
	2000	1.0851	0.0654	0.05	12.13354
	500	1.0831	0.0580	0.05	3.34822
	800	1.0837	0.0660	0.05	5.83868
0.05	1000	1.0850	0.0619	0.05	7.24772
	1500	1.0838	0.0605	0.05	10.20328
	2000	1.0852	0.0600	0.05	12.63834
	500	1.0828	0.0632	0.05	3.73042
	800	1.0835	0.0599	0.05	5.88198
0.1	1000	1.0852	0.0635	0.05	6.78648
	1500	1.0838	0.0605	0.05	9.61356
	2000	1.0573	0.0629	0.05	12.22966

Tabla 5.30: Distribución concentrada en ejes cartesianos sobre $\mathbb{T}^2$. Complejo alfa.

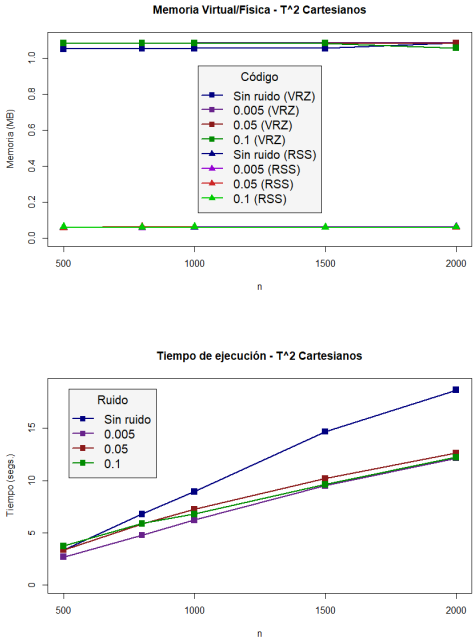


Figura 5.26: Uso de memoria física y virtual.

De las tablas 5.21-5.30 es posible observar que los 3 algoritmos mantienen un comportamiento similar a lo descrito para $\mathbb{S}^1$. Los complejos Vietoris-Rips mantienen un comportamiento polinomial, lo que concuerda con que el tiempo de ejecución es $O(n^3)$. Es posible observar tanto para $\mathbb{S}^2$ como para $\mathbb{T}^2$ el uso de memoria de los algoritmos MS y alfa se mantiene constante (sobre un 0.05 % del total). El tiempo de ejecución en el caso de MS se mantiene lineal, pero con una pendiente mucho menor comparando al caso $\mathbb{S}^1$; por otro lado, el tiempo de ejecución en los complejos alfa, mantiene un comportamiento lineal con respecto al tamaño de muesta de manera más marcada, con una pendiente y un tiempo de ejecución mayores.

Existen diferencias significativas en cuanto al tiempo de ejecución en el caso de los complejos alfa cuando se calculan sobre $\mathbb{S}^2$ y $\mathbb{T}^2$. Esto se debe a que el espacio ambiente ($\mathbb{R}^2$) debe triangularse, lo que se vuelve más costoso conforme se aumenta la dimensión del mismo.

La ejecución del algoritmo de los complejos MS sobre $\mathbb{T}^2$ no se incluyó en esta sección debido a una serie de inconsistencias al detectar la homología subyacente de los datos. El mayor costo se detectó al momento de obtener la rejilla necesaria para el cálculo de la homología, si esta era muy fina el recurso computacional con el que se contaba se agotaba. Si era muy gruesa, las propiedades topológicas no se encontraban (recuérdese que sabemos de antemano la topología subyacente). Por otro lado, es necesaria una elección óptima del ancho de banda para el estimador Kernel. De modo que se debe optimizar la selección de parámetros por dos frentes: el ancho de rejilla y el ancho de banda del estimador kernel.

Al llevar a cabo las simulaciones con las distribuciones concentradas en regiones y con repulsión en hiperplanos, nos encontramos con dificultades técnicas para realizar el análisis en el caso de las variedades $\mathbb{S}^2$ y $\mathbb{T}^2$. En el caso de los complejos VR y MS, los cálculos lograban consumir en su totalidad la memoria RAM del servidor, llegando en ocasiones a saturar la mayoría de los núcleos del procesador. Por otro lado, los complejos alfa sólo son aplicables a variedades de dimensión 2 o menor, debido a la triangulación que debe hacerse al espacio ambiente.

A pesar de esto, creemos que los cálculos que incluidos en esta sección son bastante ilustrativos para determinar el tamaño de muestra adecuado al efectuar simulaciones con estos algoritmos.

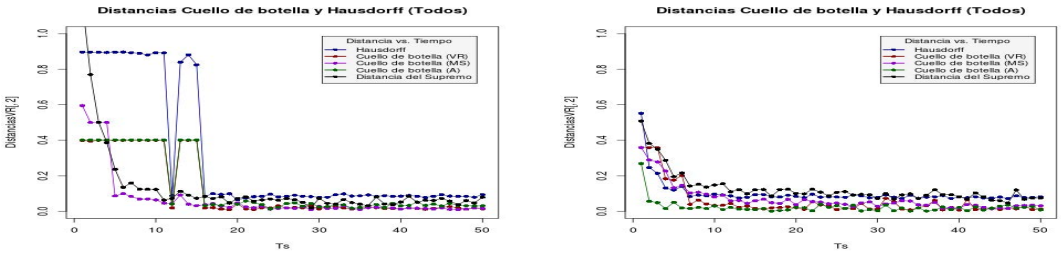
5.3. Simulaciones: estabilidad

En el Capítulo 2 introdujimos los conceptos de las distancias cuello de botella y Hausdorff ($d_B(D_1, D_2)$ y $d_H(X, Y)$, respectivamente). La primera se calcula utilizando los diagramas de persistencia D_1 y D_2 y el conjunto de biyecciones entre estos. La segunda se calcula utilizando la métrica del espacio ambiente y su objetivo es encontrar la máxima distancia existente de un conjunto a un punto más cercano en el otro. El objetivo de esta sección es analizar el comportamiento de estas dos distancias cuando se comparan las nubes de una distribución uniforme con las de una perturbación de la misma. Específicamente, en nuestros ejemplos se compara la "magnitud" de diferencia que existe en la relación que dada en el teorema de estabilidad

$$d_B(D_1, D_2) \leqslant d_H(X, Y),$$

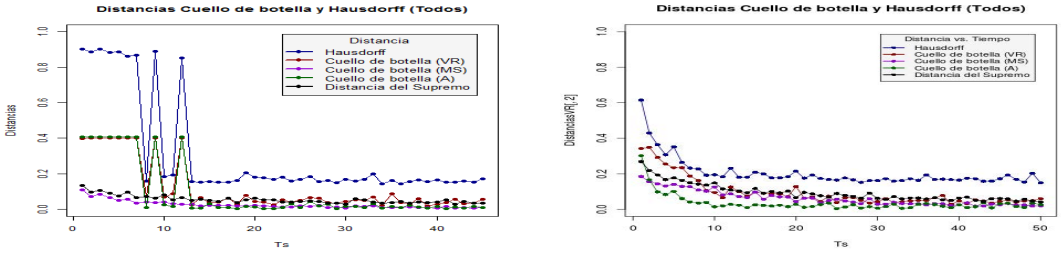
donde el lado izquierdo es la distancia entre los diagramas de persistencia respectivos.

5.3.1. Caso I: Aproximaciones a la distribución uniforme



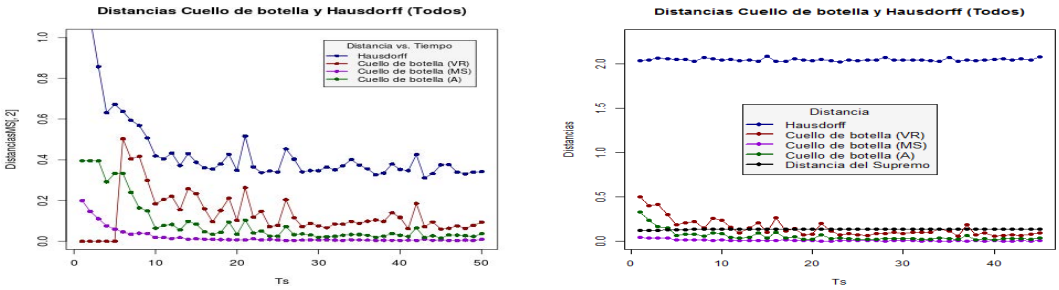
(a) Poisson - Poisson (b) Poisson - Gaussiano Inverso

Figura 5.27: Perturbaciones a la distribución uniforme sobre $\mathbb{S}^1$. En la figura izquierda, se perturba la primera y segunda entrada mediante un proceso Poisson de parámetro $\lambda = 0.2$. En la figura de la derecha, la segunda entrada mediante un proceso Gaussiano Inverso de parámetros $\mu = 0.2$ y $\lambda = 0.01$.



(a) Poisson - Poisson (b) Poisson - Gaussiano Inverso

Figura 5.28: Perturbaciones a la distribución uniforme sobre $\mathbb{S}^2$. En la figura izquierda, se perturba la primera, segunda y tercera entrada mediante un proceso Poisson de parámetro $\lambda = 0.2$. En la figura de la derecha, la segunda y tercera entrada mediante un proceso Gaussiano Inverso de parámetros $\mu = 0.2$ y $\lambda = 0.01$.



(a) Poisson - Poisson

(b) Poisson - Gaussiano Inverso

Figura 5.29: Perturbaciones a la distribución uniforme sobre $\mathbb{T}^2 = \mathbb{S}^1 \times \mathbb{S}^1$. Cada $\mathbb{S}^1$ del producto cartesiano se perturba de igual manera. En la figura izquierda, se perturba la primer y segunda entrada mediante un proceso Poisson de parámetro $\lambda = 0.2$. En la figura de la derecha, la segunda entrada mediante un proceso Gaussiano Inverso de parámetros $\mu = 0.2$ y $\lambda = 0.01$.

En las Figuras 5.27, 5.28 y 5.29 se muestra la distancia Hausdorff (Azul) calculada entre una nube de 500 puntos con distribución uniforme y una nube de 500 puntos con distribución uniforme perturbada mediante procesos Poisson de parámetro $\lambda = 0.2$ y procesos Gaussiano Inverso de parámetros $\mu = 0.2$ y $\lambda = 0.1$ (en cada figura se indica el tipo de perturbación) para tiempos de 1 a 50. Las otras curvas representan las distancias cuello de botella entre los diagramas de persistencia mediante los algoritmos Vietoris-Rips (Rojo), Morse-Smale (Violeta) y alfa (Verde). Se calcula también la distancia del supremo (Negro) entre las densidades estimadas respectivas a las dos nubes de puntos en cada tiempo.

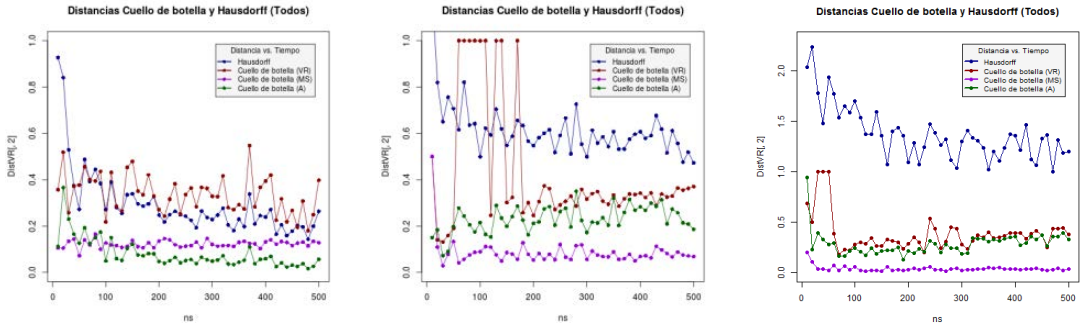
En las Figuras 5.27 y 5.28 podemos observar que el caso Poisson - Gaussiano inverso alcanza estabilidad más rápido que el caso Poisson - Poisson. Esto se debe a que en el caso que ambas perturbaciones son Poisson, dado el valor del parámetro en los primeros tiempos, se tiene una probabilidad alta de tener ceros (outliers) lo cual se ve reflejado en ambas gráficas. En ambas figuras es posible observar cómo el comportamiento de la distancia entre los diagramas de Rips sigue un comportamiento muy parecido a la distancia de Hausdorff; es por esto último que decimos que el algoritmo VR es sensible al cambio de geometría a lo largo del tiempo. En el caso de los complejos de Morse se observa que en los primeros tiempos éste presenta distancias mayores a los complejos alfa pero menores a los de Rips; esto se debe a que el algoritmo solamente calcula los puntos críticos y los diagramas contienen muy poco ruido por lo que es más sencillo para éste evaluar las distancias y presenta distancias menores. En el caso de los complejos alfa se encuentra la topología subyacente a las nubes de datos y es por esto que la estabilidad de las distancias cuello de botella se da de manera rápida.

En la Figura 5.29 es posible ver el mismo comportamiento sensible a la geometría del algoritmo Vietoris-Rips. Asímismo, el algoritmo de los complejos alfa muestra cierta sensibilidad a la geometría de los datos y la distancia cuello de botella tarda más tiempo en estabilizarse. Por otro lado, la distancia de los complejos Morse-Smale "converge" casi inmediatamente a cero. Creemos que esto se debe a la manera que se están simulando las

nubes (sobre $\mathbb{R}^4$ y proyectando luego a $\mathbb{R}^3$) pues posiblemente la proyección que utilizamos nos lleve a cierta "pérdida de información" que el algoritmo es incapaz de detectar.

5.3.2. Caso II: Distribución con repulsión vs. distribución uniforme

Las siguientes gráficas muestran una comparación entre las distancias cuello de botella para distintos algoritmos y las distancias de Hausdorff entre nubes de puntos simuladas con distribución con repulsión y distribución uniforme, para distintos tamaños de muestra.



(a) Distancias en $\mathbb{S}^1$-repulsión. (b) Distancias en $\mathbb{S}^2$-repulsión. (c) Distancias en $\mathbb{T}^2$-repulsión.

En esta última figura, es posible ver como la estabilidad no siempre se satisface para VR. Esto puede deberse a la diferencia "física" existente entre las dos nubes de datos que estamos comparando, así como a la "complejidad" de la variedad en la que se está comparando. Con esto reafirmamos que el algoritmo VR es sensible a la geometría de los datos.

Por otro lado, vemos que las distancias cuello de botella de los diagramas de los complejos MS y alfa siempre son estables respecto a la distancia de Hausdorff entre las nubes de datos que estamos comparando. En ambos casos podemos hablar de la robusticidad de los algoritmos para detectar la topología subyacente de los datos.

En el Apéndice B se muestra la comparación de la distancia de Hausdorff entre nubes de datos con las aproximaciones a la distribución uniforme vs. la distribución uniforme, así como la distancia del supremo entre las densidades estimadas asociadas a los mismos. Igualmente, se incluye esta comparación en el caso de la distribución con repulsión vs. la distribución uniforme.

Asimismo, en el Apéndice B se presentan numerosas simulaciones de nubes de datos con distribución con repulsión en las variedades $\mathbb{S}^1$, $\mathbb{S}^2$ y $\mathbb{T}^2$ y los diagramas de persistencia respectivos a los algoritmos VR, MS y alfa. El objetivo de estas simulaciones es ilustrar la efectividad de estos algoritmos para detectar la topología subyacente, así como comparar de manera visual diagramas de persistencia asociados a esta distribución.

6

Conclusiones y recomendaciones

En este capítulo se presentan las conclusiones de los estudios de simulación comparativos de algoritmos que se realizaron en este trabajo, así como características que se experimentaron en el uso de estos algoritmos. Esperamos que éstas puedan servir como referencia a aquellos que se inician en el cálculo de la homología persistente.

6.1. Conclusiones generales

Los resultados obtenidos en la Sección 5.1 pueden servir como referencia a cualquier usuario que desee utilizar los algoritmos de persistencia que presentamos en esta tesis. De las tablas presentadas se puede saber si nuestro equipo de cómputo es suficiente para realizar cálculos de homología e incluso qué tamaño de muestra o distribución podemos simular.

Con relación a robusticidad de algoritmos respecto a la distribución en las variedades consideradas, de las simulaciones realizadas a lo largo de esta tesis podemos concluir lo siguiente. En general los algoritmos detectan la topología subyacente de las nubes de datos simuladas con distribución cercana a la distribución uniforme. Lo que se observa relevante es contar con una muestra de buen tamaño y lo suficientemente densa sobre la variedad. Sin embargo, los algoritmos son sensibles a la geometría y distribución de las nubes de puntos, como se explica en las conclusiones globales al final de este capítulo.

De las Figuras en la Sección 5.3 se perciben dos hechos. Primero, para las nubes de datos de las aproximaciones a la distribución uniforme en las variedades consideradas se satisface la estabilidad de la distancia entre los diagramas de persistencia de los algoritmos cuando el tiempo de la aproximación crece, respecto a la distancia de Hausdorff entre las nubes de puntos cuando se tiene un tamaño de muestra moderado. Esta estabilidad no se obtiene cuando el tiempo de la aproximación es pequeño, pues se cuenta con demasiada dispersión en los datos. Sin embargo, no necesariamente se satisface este teorema respecto

Algoritmo	Geometría	Topología	Distribución	Dimensión Máxima	Paquete
Rips	✓	✓	✗	3	TDA-R
Morse	✗	✓	✓	2	TDA-R
Alfa	✗	✓	✗	2	TDA-R
Testigo	✗	✓	✗	5	JPlex
Mapper	✗	✓	✓	1	Python

Tabla 6.1: El apartado de **Dimensión máxima** se refiere a la dimensión homológica soportada por el software (sin colapsar) en el servidor utilizado para las simulaciones.

a la distancia del supremo entre las densidades estimadas a partir de las nubes de datos.

Por otro lado, para distribuciones distintas a la uniforme como la distribución con repulsión en hiperplanos se satisface la estabilidad de la distancia cuello de botella entre los diagramas de persistencia de las nubes de datos con respecto a la distancia de Hausdorff, en el caso de $\mathbb{S}^2$ para $n > 200$ y para $\mathbb{T}^2$ aún con muestras pequeñas, pero esto no se cumple para $\mathbb{S}^1$. Sin embargo, las distancias cuello de botella en todos estos casos no es pequeña, exceptuando el algoritmo MS. Esto es congruente con la visualización de los diagramas de persistencia mostrados en el Apéndice B. Esto indica la no necesaria cercanía entre los diagramas de persistencia correspondientes lo cual está en contraste a lo que se observa cuando se tienen distribuciones cercanas a la uniforme. La explicación de esto es que la distribución con repulsión genera al menos dos componentes conexas en las nubes de datos dependiendo la variedad, con lo que no se cumple que los datos sean suficientemente densos en toda la variedad.

Dada la naturaleza de la construcción de los complejos alfa, estos son robustos ante el cambio de distribución en las distintas variedades. Esto se debe posiblemente a que se utilizan distintos pesos para los puntos de la nube dependiendo qué tan densas sean sus vecindades. Por otro lado, los valores de filtración máximos en los complejos MS son pequeños, por lo que las distancias cuello de botella obtenidas entre los diagramas de persistencia de las variedades estudiadas tienden a ser pequeñas. Es posible ver en las figuras de la Sección 5.3 y el Apéndice B tanto la robusticidad de ambos algoritmos así como información a cerca de la concentración de datos en el caso del algoritmo MS.

Finalmente, en la Tabla 6.1 incluimos un resumen de las distintas características de cada uno de los algoritmos utilizados en esta tesis. Indicamos si el algoritmo de cada complejo es capaz de identificar la topología, la geometría y la distribución de los datos en las variedades trabajadas. También se incluye el nombre del paquete así como el lenguaje en el que se encuentra implementado cada algoritmo.

6.2. Conclusiones particulares

Presentamos a continuación observaciones particulares sobre cada uno de los algoritmos para posteriormente hacer conclusiones generales.

6.2.1. Algoritmo Vietoris-Rips

1. Este algoritmo determina la geometría y la topología de la nube de datos pero no la distribución de los puntos en la variedad. Esto se debe a la "sensibilidad" que tiene el algoritmo dada la fuerte relación que tienen los complejos VR con el complejo de Čech. Los resúmenes implementados para este algoritmo son los diagramas de persistencia y los códigos de barra.

2. Cuando las distribuciones sobre las variedades $\mathbb{S}^2$ y $\mathbb{T}^2$ están "rotas" (casos como normales con correlación fuerte, repulsión en hiperplanos, aproximaciones de alta concentración en tiempos de perturbación pequeños) el algoritmo no siempre puede calcular la homología si la nube de datos tiene una alta concentración de puntos. Decimos que no puede en el sentido de que agota por completo los recursos del servidor. Este comportamiento se debe a la manera en que el algoritmo construye los complejos simpliciales, generando simplejos de muy alta dimensión cuando hay regiones con grandes concentraciones de datos.

3. Es necesario conocer los valores mínimo y máximo en cada dimensión de la nube de datos, con la finalidad de poder establecer un valor máximo de filtración. Realizar este ajuste de manera correcta ayuda al algoritmo a detectar características topológicas relevantes. Se requiere una heurística adecuada y flexible para obtener o aproximar este ajuste de manera correcta.

4. Para el círculo $\mathbb{S}^1$ y la esfera $\mathbb{S}^2$, un valor de filtración máximo igual a 0.8 es suficiente para capturar de manera correcta su homología. De hecho, se puede incrementar este valor sin ningún problema, esto debido a la dimensión de la variedad.

5. El valor máximo que se puede poner a la filtración para el cálculo de la homología en un toro $\mathbb{T}^2$ debe ser al menos 1.2 para que se logren calcular los números de Betti. Sin embargo, el tamaño de muestra y la distribución en esta variedad tienen un papel importante en el uso de recursos computacionales.

6. Al estar implementado en R, el usuario puede no darse cuenta que el algoritmo se está sobre extendiendo en el uso de recursos. Esto puede llevar a un colapso de sistema e incluso perder trabajo no guardado.

7. Es necesario optimizar el uso de recursos en el algoritmo. La ejecución se debe detener cuando el proceso está haciendo uso total de los recursos del equipo de cómputo.

6.2.2. Algoritmo Morse-Smale

1. El algoritmo sólo trabaja con datos euclidianos, pues se hace uso del estimador de densidad vía kernel. Además es necesario crear una rejilla de valores en el espacio ambiente para poder evaluar el estimador de la densidad. Los resúmenes implementados para este algoritmo son los diagramas de persistencia y los códigos de barra.

2. En los diagramas de persistencia aparecen no solamente los números de Betti, sino también los puntos críticos encontrados en la densidad estimada con la nube de

datos. Por esta razón la interpretación de su diagrama de persistencia es distinta a la que se les da a los otros algoritmos y por ello la distancia cuello de botella no es comparable con sus respectivas distancias de los diagramas de persistencia de éstos.

3. Es un buen referente para encontrar puntos de alta concentración en la nube de puntos. Esto debido al estimador de densidades utilizado en la construcción del complejo simplicial.

4. La implementación realizada por Zomorodian (2005) se hace para variedades a lo más de dimensión 2 inmersas en $\mathbb{R}^3$.

5. De esta última observación, un problema principal con el toro puede deberse a que nuestra construcción se hace en $\mathbb{R}^4$ ($\mathbb{S}^1 \times \mathbb{S}^1$) pero se proyecta a $\mathbb{R}^3$.

6. Para dimensiones homológicas 0, 1 y 2, el algoritmo hace uso eficiente de los recursos computacionales.

7. El algoritmo requiere de una rejilla para el cálculo de la homología. Si ésta es muy fina resulta muy costoso computacionalmente y si es muy burda no captura las propiedades topológicas de manera correcta. Es necesario saber cómo calibrar este parámetro para lo cual es necesario conocer los límites inferiores y superiores del rango que ocupan los datos en el espacio ambiente. En el caso de datos reales llevar a cabo dicha calibración puede volverse complicado, pues posiblemente es necesario tener conocimiento de la topología subyacente a la nube de datos.

8. El uso de recursos es constante sin importar la dimensionalidad o el tamaño de la muestra. El tiempo de ejecución es lineal en el caso de $\mathbb{S}^1$ con respecto al tamaño de muestra, se hace constante en el caso de las variedades de dimensión 2 como son $\mathbb{S}^2$ y $\mathbb{T}^2$. Estas afirmaciones pueden verse en las tablas 5.11-5.15, 5.23 y 5.24.

9. Es complicado para el algoritmo encontrar la homología correcta cuando el ancho de banda y el ancho de la rejilla no están calibrados correctamente.

10. Si no se tiene conocimiento previo de la estructura geométrica de los datos se puede tornar difícil la calibración del ancho de banda y la finitud de la rejilla.

11. Conforme aumenta la dimensión en la homología, es mucho más complicado para el algoritmo calcularla de manera correcta. Posiblemente se deba a la complejidad de la rejilla. Como caso particular mencionamos el toro $\mathbb{T}^2$ que presentamos como ejemplo en el Capítulo 2.

12. No es necesario establecer un valor máximo de filtración.

13. Al igual que el algoritmo VR, puede sobre extenderse en el uso de recursos computacionales y llevar al equipo a un colapso cuando la dimensión de la nube de datos es mayor a 3.

6.2.3. Algoritmo complejos alfa

1. Es capaz de detectar únicamente la geometría de la variedad pero no la distribución de los datos en ésta, similar al caso de los complejos VR y testigo. El único resumen disponible para este algoritmo son los diagramas de persistencia.

2. Como el caso de los complejos de MS y Mapper, el algoritmo trabaja únicamente con datos euclideos. Este hecho se debe a que se realiza una triangulación del espacio ambiente, en este caso $\mathbb{R}^2$ y $\mathbb{R}^3$, y es necesario establecer un valor máximo de filtración

3. Es eficiente para encontrar la topología subyacente a los datos. En el caso de los ejemplos que ilustramos en esta tesis no toma en cuenta los *outliers* (o falta de puntos en la variedad como en los primeros tiempos de las aproximaciones a la distribución uniforme de los casos con procesos Poisson).

4. El algoritmo trabaja hasta dimensión homológica 2, debido a la complejidad de las teselaciones de Voronoi del espacio ambiente.

5. El cálculo de la homología es bastante rápido hasta dimensión homológica 2 (dimensión 3 del espacio ambiente), comparado con el resto de los algoritmos. Es posible llevarlo a cabo en un equipo de cómputo de uso personal.

6. El costo computacional del algoritmo es similar al de los complejos MS. La diferencia con estos últimos es que el tiempo de ejecución es mucho menor como lo podemos ver en las tablas 5.16-5.20, el cual también muestra un comportamiento lineal con respecto al tamaño de muestra.

7. No detecta $1-$ciclos cuando existen *outliers* en el caso de las perturbaciones a la distribución uniforme en la variedad. Esto en contraste con los complejos testigo y VR.

8. La "limpieza"de los diagramas de persistencia se debe a la maleabilidad que tiene el algoritmo al construir las vecindades alrededor de puntos que tengan alta concentración en la variedad.

9. Debido a lo reciente de la implementación del algoritmo en la librería TDA de R, es necesario modificar el valor (Inf) que devuelve en el caso de las componentes conexas que persisten durante toda la filtración para que estos se puedan mostrar en el diagrama de persistencia.

6.2.4. Algoritmo complejos Testigo

1. Al igual que el algoritmo VR, este algoritmo es eficaz en detectar la geometría de la nube de datos pero no logra capturar la distribución de la misma. Cuenta solamente con los códigos de barra como resumen topológico.

2. Es fácil "engañarlo" si la nube de datos tiene *outliers*. Esto hace necesario preparar los datos antes de analizarlos haciendo limpieza de datos "extraños".

3. Una selección pobre de puntos de referencia nos lleva a una estimación pobre de los números de Betti. Si la cantidad de puntos seleccionados es muy grande, el algoritmo empieza a detectar agujeros y componentes que no existen en la realidad. La sugerencia de De Silva y Carlsson (2004) de tomar k puntos de modo que se tenga una razón $n/k \leqslant 20$ funcionó de buena manera en los casos que consideramos.

4. Es necesario calibrar el valor máximo de la filtración. Un buen comienzo es tomar como parámetro a R, la distancia máxima que existe entre los puntos de referencia y el resto de la nube de puntos. A dicho valor se le toma una proporción (regularmente tomamos $R/4$ o $R/5$) y a partir de aquí hacerlo crecer cuidando de que no se sature la memoria.

5. La implementación está desarrollada en Matlab, lo que ayuda a que el uso de recursos sea eficiente. Si el algoritmo se cicla o se prevé falta de recursos, envía un mensaje al usuario que permite recalibrar los parámetros.

6.2.5. Algoritmo Mapper

1. Es efectivo como auxiliar para encontrar las componentes conexas que en ocasiones pasan desapercibidas para los algoritmos VR y MS. Los algoritmos de agrupamiento ayudan en esta situación.

2. Ayuda a localizar concentraciones de puntos en la nube de datos. Esto se describe en el tamaño y coloración de los vértices en el $1-$simplejo generado.

3. Cuando existen *outliers*, el algoritmo los detecta como un grupo aparte y no los mezcla con la geometría de la "nube principal" de datos.

4. Al igual que el algoritmo Morse-Smale, este algoritmo trabaja solamente con datos euclideos. Los datos se deben introducir al software en forma matricial y sin encabezados.

5. El algoritmo no tiene problemas con el uso de recursos. De hecho es bastante eficiente y rápido. Se puede utilizar en cualquier equipo de uso personal.

6. Con la implementación actual que es libre, el algoritmo sólo puede detectar agujeros de dimensión 0 y 1. Sin embargo, es un buen complemento para los otros algoritmos pues nos ayuda a tener idea de la estructura de los datos de una manera rápida.

7. El comportamiento con los filtros implementados en Python Mapper es similar para el cálculo del complejo mediante este algoritmo, como se ilustró en la Sección 4.3.

6.3. Conclusiones globales

1. De las Figuras 5.27, 5.28 y 5.29 se puede observar que el algoritmo VR es el más sensible en cuanto a cambios de la geometría de los datos, pues la distancia cuello de botella de éstos siguen un patrón similar al de la distancia Hausdorff. Por otra parte, el algoritmo de los complejos alfa es para el que la distancia de sus diagramas "cae"

más rápido a cero. Esto último se debe a que el algoritmo es solamente sensible a la topología subyacente de la nube de datos y debido a la rapidez de su cálculo no detecta geometría o distribución en las nubes de datos. De los ejemplos del Capítulo 2 se observa que el algoritmo de los complejos de Morse-Smale es bastante sensible a la distribución de los datos, pero en la gráfica de distancias en la Figura 5.29 vemos cómo la distancia entre los diagramas de persistencia cae rápidamente. Esto puede deberse a que los diagramas de persistencia capturan mucha información "errónea" como puede verse en las Figuras de la Sección 2.6.3.

2. Es necesario optimizar el algoritmo de limpieza de los datos sugerido por Bubenik et al. (2010). Este algoritmo se puede utilizar de manera previa al cálculo de la homología. Esto con la finalidad de no "confundir" a los algoritmos y el cálculo de la homología sea más preciso.

3. Si solamente deseamos capturar la geometría o la topología de nuestras nubes de datos, es necesario "limpiar" las altas concentraciones de puntos. Queda como trabajo futuro una correcta implementación de esta idea.

4. El algoritmo VR es el mejor para detectar la geometría y topología de las nubes datos debido a su cercanía con el complejo de Čech. Se recomienda utilizarlo si se cuenta con un buen recurso computacional, pero monitoreando el uso de este recurso. Se pueden tomar como referencia las tablas de la Sección 5.2.

5. Existen algoritmos que hacen uso de vecindades ponderadas en los algoritmos VR y alfa que se comportan de una forma que se adapta a la densidad de los puntos. Esto tiene repercusión en una correcta detección de las propiedades topológicas subyacentes a nubes de puntos dadas. Por ejemplo en el trabajo de Dey et al. (2016) se utilizan además colapsos de simplejos.

6. De las tablas presentadas en la Sección 5.1 podemos concluir que cuando fijamos una distribución y aumentamos el tamaño de ruido, el costo computacional y el tiempo de ejecución disminuyen.

7. En las distribuciones "parecidas" a la uniforme como lo son la concentrada en ejes cartesianos y la concentrada por regiones (cuando la correlación de las variables es chica) el cálculo de los diagramas de persistencia tiene un costo y tiempo de ejecución muy parecido. Esto último se debe a que las nubes de datos presentan cierta "continuidad" a diferencia de la distribución concentrada en regiones con alta correlación y la distribución con repulsión en hiperplanos. En estos casos el cálculo se vuelve más costoso e incluso no se logra concluir en las variedades de dimensión 2.

8. Continuando con la referencia a las tablas de la Sección 5.1, es posible observar que el costo computacional (memoria virtual y física) de los complejos MS se mantiene constante y el tiempo es lineal en el caso de $\mathbb{S}^1$.

9. El algoritmo Mapper es el más rápido en tiempo de ejecución, se puede instalar y utilizar en cualquier equipo de cómputo personal.

10. Los algoritmos MS y Mapper son buenos para detectar distribución de los datos (concentraciones en regiones y efectos de repulsión). El costo computacional de estos algoritmos es pequeño en dimensiones bajas, comparados con el algoritmo VR.

11. El algoritmo de los complejos alfa es bastante eficiente en tiempo de ejecución y uso de recursos. Además encuentra de manera eficiente la topología subyacente de las nubes de datos que trabajamos.

12. Hace falta encontrar una manera correcta de estimar la cantidad de puntos de referencia que utiliza el algoritmo de los complejos testigo. Si se tienen muy pocos, captura muy pocas propiedades reales de los datos. Si se tienen muchos puntos de referencia, se introduce mucho ruido en las características que se detectan en los códigos de barra del algoritmo.

13. Un proyecto de interés personal, es realizar la implementación del algoritmo Mapper para dimensiones mayores a 1. También, llevar a cabo la implementación propuesta por Hennigan (vista en el Capítulo 4).

14. En cuanto a los tres algoritmos estudiados en el Capítulo 5, se debe elegir dependiendo lo que se desee: precisión o velocidad. Lo primero se obtiene al obtener un cálculo lo más aproximado a la realidad utilizando el complejo VR y lo segundo se obtiene haciendo una elección entre MS o alfa, dependiendo la complejidad de la topología subyacente de los datos. Dicha complejidad se puede detectar en los tiempos de ejecución y uso de recursos de los complejos MS o alfa, pues si el cálculo se vuelve costoso es que algo va mal con los datos.

15. Es necesario tener una buena documentación para la instalación de los distintos algoritmos de ATD, de manera que su instalación sea sencilla para cualquier persona interesada.

A

Algoritmos de aproximación a esferas

En este apéndice mostramos tres algoritmos para simular datos con perturbaciones a la distribución uniforme en una $d-$esfera $\mathbb{S}^d$. El primero perturba las entradas mediante un proceso Poisson de parámetro λ, el segundo las perturba mediante un proceso inverso gaussiano de parámetros α y β. El parámetro α es la media y β es el parámetro de forma de una distribución inversa gaussiana. Las perturbaciones en el último caso son por medio de un proceso normal inverso gaussiano de parámetros α, β y δ. El parámetro α indica lo pesado de las colas de la distribución, β indica la simetría y δ es el parámetro de escala.

En todos los casos se pide el tamaño n de la muestra a simular y el tiempo t en el que se quiere realizar la perturbación. El parámetro d es la dimensión de la esfera que se desea simular.

Algorithm 1 Algoritmo: Aproximación en $\mathbb{S}^d$ mediante perturbación por proceso Poisson

function SD.PP(n,t,d,λ)
  $S \leftarrow 0_{n\times d}$
  for $i \leftarrow 1$ to n **do**
    $m \leftarrow$ Cuantil Poisson(λt) de un valor cercano a 1, en este caso $1 - 1x10^{-12}$
    $X \leftarrow 0_{d\times m}$
    $P \leftarrow 0_{d\times m}$
    $\sigma \leftarrow 0_d$
    for $j \leftarrow 1$ to d **do**
      $X_{i,\cdot} \leftarrow (\xi_1, \xi_2, ..., \xi_m)$ ▷ $\xi_s \sim$ Exp(λ)
      $P_{i,\cdot} \leftarrow (\xi_1, \xi_1 + \xi_2, \xi_1 + \xi_2 + \xi_3, ..., \xi_1 + \xi_2 + \cdots + \xi_m)$
      $\sigma_i \leftarrow \sum_{k=1}^{m} \mathbb{1}_{P_{i,k}<t}$
    if $\sigma_i! = 0$ para algún i **then**
      $S_{i,\cdot} \leftarrow \sigma_1 Y_1, \sigma_2 Y_2, ..., \sigma_d Y_d)$ ▷ $Y_i \sim N(0, 1)$
      $S_{i,\cdot} \leftarrow (\frac{S_{1,\cdot}}{\|S_{1,\cdot}\|}$
    else
      $S_{i,\cdot} \leftarrow 0_d$
  return S

Algorithm 2 Algoritmo: Aproximación en $\mathbb{S}^d$ mediante perturbación por proceso inverso gaussiano

function SD.IG(n,t,d,μ,λ)
  $S \leftarrow 0_{n\times d}$
  for $i \leftarrow 1$ to n **do**
    $\sigma \leftarrow 0_d$
    for $i \leftarrow 1$ to d **do**
      $aux \leftarrow (I_1, I_1 + I_2, I_1 + I_2 + I_3, ..., I_1 + I_2 + \cdots + I_t)$ ▷ $I \sim$ IG(μ, λ)
      $\sigma_i \leftarrow aux_t$
    $S_{i,\cdot} \leftarrow (\sigma_1 Y_1, \sigma_2 Y_2, ..., \sigma_d Y_d)$ ▷ $Y_i \sim N(0, 1)$
    $S_{i,\cdot} \leftarrow \frac{S_{1,\cdot}}{\|S_{1,\cdot}\|}$
  return S

Algorithm 3 Algoritmo: Aproximación en $\mathbb{S}^d$ mediante perturbación por proceso normal inverso gaussiano

function SD.IG(n,t,d,δ,α,β)
  $S \leftarrow 0_{n\times d}$
  for $i \leftarrow 1$ to$n$ **do**
    $\sigma \leftarrow 0_d$
    for $i \leftarrow 1$ to d **do**
      $I \leftarrow (0,0)$
      $I_1 \leftarrow 0$
      $I_2 \leftarrow I_1 + Z$ ▷ $Z \sim \text{IG}(1, \delta\sqrt{\alpha^2-\beta^2})$
      $W \leftarrow 0$
      for $i \leftarrow 2$ to t **do**
        $dt \leftarrow I_2 - I_1$
        $W \leftarrow W + \sqrt{dt}\cdot Y$ ▷ $Y \sim N(0,1)$
        $N \leftarrow \beta\delta^2 + I_2 + \delta W$
        $I \leftarrow (I_2, I_2 + Z)$ ▷ $Z \sim \text{IG}(1, \delta\sqrt{\alpha^2-\beta^2})$
      $\sigma_i \leftarrow N$
    $S_{i,\cdot} \leftarrow (\sigma_1 Y_1, \sigma_2 Y_2, ..., \sigma_d Y_d)$ ▷ $Y_i \sim N(0,1)$
    $S_{i,\cdot} \leftarrow \frac{S_{1,\cdot}}{\|S_{1,\cdot}\|}$
  return S

B

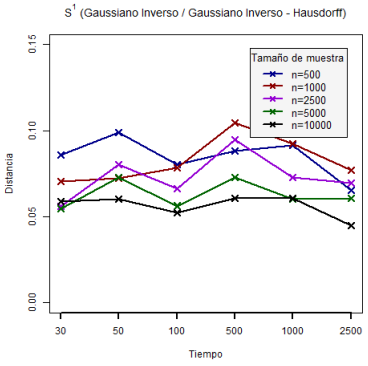
Distancias entre distribuciones y entre nubes de datos simuladas

En este apéndice se muestra la comparación de la distancia de Hausdorff entre nubes de datos con las aproximaciones a la distribución uniforme vs. la distribución uniforme, así como la distancia del supremo entre las densidades estimadas asociadas a los mismos. Se incluye también esta comparación en el caso de la distribución con repulsión vs. la distribución uniforme.

Como complemento a las simulaciones de los Capítulos 2, 3 y 5, también se presentan numerosas simulaciones de nubes de datos con distribución con repulsión en las variedades $\mathbb{S}^1$, $\mathbb{S}^2$ y $\mathbb{T}^2$ y los diagramas de persistencia respectivos a los algoritmos VR, MS y Alfa.

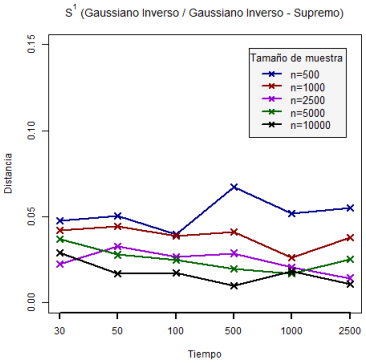
Poisson / Poisson (Hausdorff)						
n \ t	**30**	**50**	**100**	**500**	**1000**	**2500**
500	0.0859	0.0990	0.0803	0.0883	0.0912	0.0653
1000	0.0704	0.0722	0.0785	0.1047	0.0924	0.0771
2500	0.0561	0.0801	0.0664	0.0946	0.0729	0.0697
5000	0.0548	0.0729	0.0563	0.0727	0.0602	0.0608
10000	0.0588	0.0604	0.0522	0.0605	0.0609	0.0449

Tabla B.1: Distancia Hausdorff para perturbaciones a la distribución uniforme en $\mathbb{S}^1$ mediante procesos Poisson $N_1(t)$ y $N_2(t)$ de parámetro $\lambda = 0.2$.



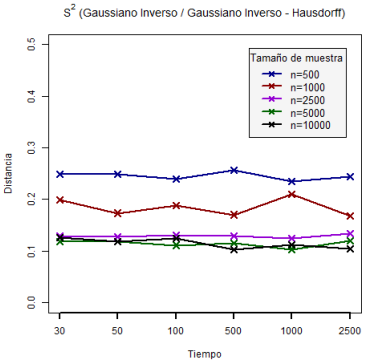
Poisson / Poisson (Supremo)						
n \ t	**30**	**50**	**100**	**500**	**1000**	**2500**
500	0.0475	0.0506	0.0398	0.0670	0.0518	0.0552
1000	0.0419	0.0446	0.0388	0.0412	0.0262	0.0380
2500	0.0227	0.0327	0.0267	0.0287	0.0208	0.0143
5000	0.0368	0.0278	0.0249	0.0198	0.0170	0.0253
10000	0.0290	0.0171	0.0173	0.0099	0.0180	0.0110

Tabla B.2: Distancia del Supremo para perturbaciones a la distribución uniforme en $\mathbb{S}^1$ mediante procesos Poisson $N_1(t)$ y $N_2(t)$ de parámetro $\lambda = 0.2$.



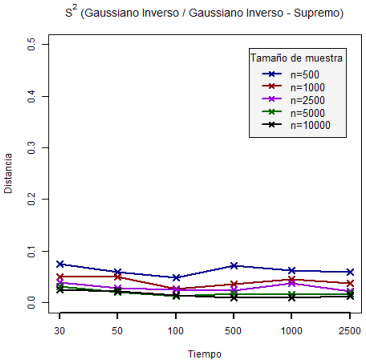
Poisson / Poisson (Hausdorff)						
n \ t	**30**	**50**	**100**	**500**	**1000**	**2500**
500	0.2482	0.2486	0.2399	0.2561	0.2353	0.2440
1000	0.1999	0.1735	0.1880	0.1704	0.2096	0.1676
2500	0.1285	0.1282	0.1311	0.1299	0.1242	0.1343
5000	0.1178	0.1187	0.1114	0.1150	0.1036	0.1198
10000	0.1263	0.1180	0.1252	0.1035	0.1118	0.1051

Tabla B.3: Distancia Hausdorff para perturbaciones a la distribución uniforme en $\mathbb{S}^1$ mediante procesos Poisson $N_1(t)$, $N_2(t)$ y $N_3(t)$ de parámetro $\lambda = 0.2$.



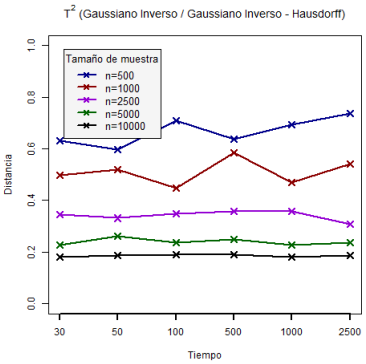
Poisson / Poisson (Supremo)						
n \ t	**30**	**50**	**100**	**500**	**1000**	**2500**
500	0.0743	0.0590	0.0481	0.0712	0.0621	0.0589
1000	0.0501	0.0502	0.0263	0.0365	0.0447	0.0370
2500	0.0391	0.0280	0.0249	0.0244	0.0382	0.0218
5000	0.0312	0.0208	0.0127	0.0175	0.0162	0.0181
10000	0.0255	0.0228	0.0144	0.0098	0.0102	0.0135

Tabla B.4: Distancia del Supremo para perturbaciones a la distribución uniforme en $\mathbb{S}^1$ mediante procesos Poisson $N_1(t)$, $N_2(t)$ y $N_3(t)$ de parámetro $\lambda = 0.2$.



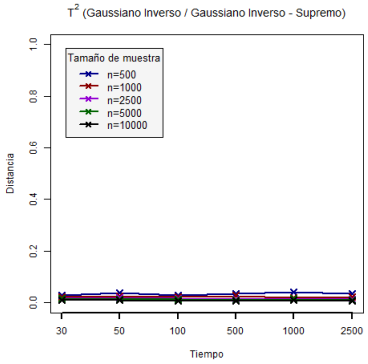
Poisson / Poisson (Hausdorff)						
n \ t	**30**	**50**	**100**	**500**	**1000**	**2500**
500	0.6301	0.5979	0.7101	0.6365	0.6934	0.7365
1000	0.4981	0.5206	0.4470	0.5844	0.4687	0.5415
2500	0.3456	0.3317	0.3479	0.3591	0.3583	0.3088
5000	0.2262	0.2614	0.2375	0.2502	0.2269	0.2367
10000	0.1820	0.1867	0.1915	0.1899	0.1811	0.1859

Tabla B.5: Distancia Hausdorff para perturbaciones a la distribución uniforme en $\mathbb{S}^1$ mediante procesos Poisson $N_1(t)$, $N_2(t)$, $N_3(t)$ y $N_4(t)$ de parámetro $\lambda = 0.2$.



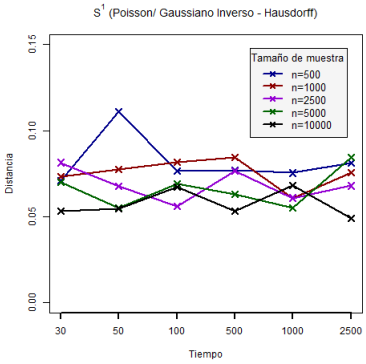
Poisson / Poisson (Supremo)						
n \ t	**30**	**50**	**100**	**500**	**1000**	**2500**
500	0.0287	0.0376	0.0287	0.0341	0.0412	0.0339
1000	0.0218	0.0256	0.0227	0.0268	0.0184	0.0217
2500	0.0174	0.0202	0.0157	0.0120	0.0162	0.0119
5000	0.0151	0.0132	0.0117	0.0102	0.0144	0.0095
10000	0.0100	0.0092	0.0070	0.0069	0.0088	0.0078

Tabla B.6: Distancia del Supremo para perturbaciones a la distribución uniforme en $\mathbb{S}^1$ mediante procesos Poisson $N_1(t)$, $N_2(t)$, $N_3(t)$ y $N_4(t)$ de parámetro $\lambda = 0.2$.



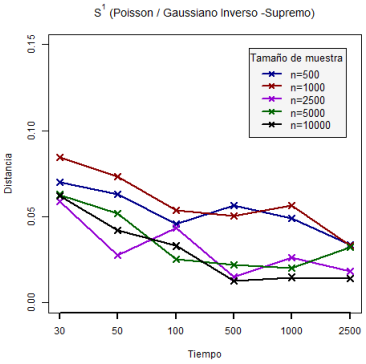
Poisson / Gaussiano Inverso (Hausdorff)						
n \ t	**30**	**50**	**100**	**500**	**1000**	**2500**
500	0.0707	0.1109	0.0764	0.0769	0.0754	0.0810
1000	0.0732	0.0774	0.0818	0.0843	0.0605	0.0757
2500	0.0814	0.0675	0.0559	0.0763	0.0609	0.0679
5000	0.0701	0.0549	0.0690	0.0630	0.0550	0.0843
10000	0.0533	0.0544	0.0672	0.0534	0.0682	0.0489

Tabla B.7: Distancia Hausdorff para perturbaciones a la distribución uniforme en $\mathbb{S}^1$ mediante procesos Poisson $N_1(t)$ e $\text{IG}_2(t)$ de parámetros $\lambda_P = 0.2$ y $\mu_{IG} = 0.2$, $\lambda_{IG} = 0.01$.



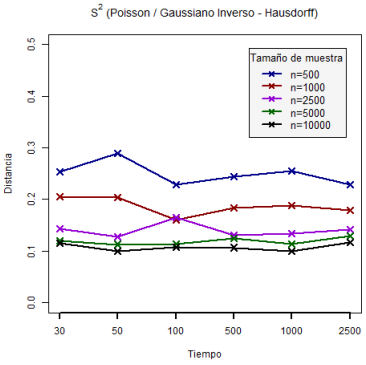
Poisson / Gaussiano Inverso (Supremo)						
n \ t	**30**	**50**	**100**	**500**	**1000**	**2500**
500	0.0698	0.0632	0.0456	0.0564	0.0489	0.0338
1000	0.0845	0.0732	0.0538	0.0506	0.0564	0.0333
2500	0.0588	0.0274	0.0436	0.0148	0.0261	0.0182
5000	0.0628	0.0517	0.0254	0.0219	0.0202	0.0322
10000	0.0619	0.0419	0.0330	0.0126	0.0148	0.0140

Tabla B.8: Distancia del Supremo para perturbaciones a la distribución uniforme en $\mathbb{S}^1$ mediante procesos Poisson $N_1(t)$ e $\text{IG}_2(t)$ de parámetros $\lambda_P = 0.2$ y $\mu_{IG} = 0.2$, $\lambda_{IG} = 0.01$.



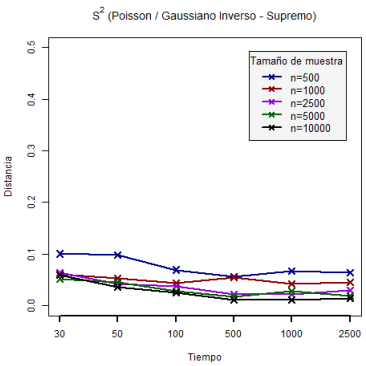
Poisson / Gaussiano Inverso (Hausdorff)						
n \ t	**30**	**50**	**100**	**500**	**1000**	**2500**
500	0.2540	0.2894	0.2287	0.2442	0.2545	0.2292
1000	0.2053	0.2033	0.1600	0.1832	0.1890	0.1786
2500	0.1430	0.1275	0.1656	0.1306	0.1343	0.1413
5000	0.1205	0.1129	0.1133	0.1242	0.1134	0.1291
10000	0.1156	0.1002	0.1078	0.1067	0.0999	0.1170

Tabla B.9: Distancia Hausdorff para perturbaciones a la distribución uniforme en $\mathbb{S}^1$ mediante procesos Poisson $N_1(t)$, $\text{IG}_2(t)$ e $\text{IG}_3(t)$ de parámetros $\lambda_P = 0.2$ y $\mu_{IG} = 0.2$, $\lambda_{IG} = 0.01$.



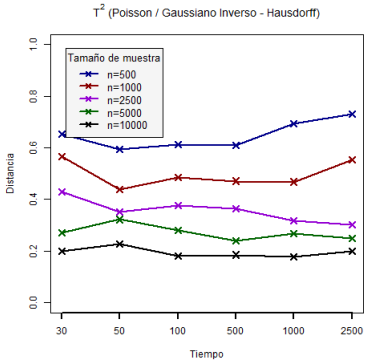
Poisson / Gaussiano Inverso (Supremo)						
n \ t	**30**	**50**	**100**	**500**	**1000**	**2500**
500	0.1005	0.0979	0.0694	0.0560	0.0665	0.0647
1000	0.0607	0.0525	0.0439	0.0543	0.0424	0.0454
2500	0.0636	0.0421	0.0381	0.0214	0.0221	0.0303
5000	0.0523	0.0454	0.0281	0.0175	0.0278	0.0187
10000	0.0594	0.0366	0.0258	0.0115	0.0114	0.0139

Tabla B.10: Distancia del Supremo para perturbaciones a la distribución uniforme en $\mathbb{S}^1$ mediante procesos Poisson $N_1(t)$, $\text{IG}_2(t)$ e $\text{IG}_3(t)$ de parámetros $\lambda_P = 0.2$ y $\mu_{IG} = 0.2$, $\lambda_{IG} = 0.01$.



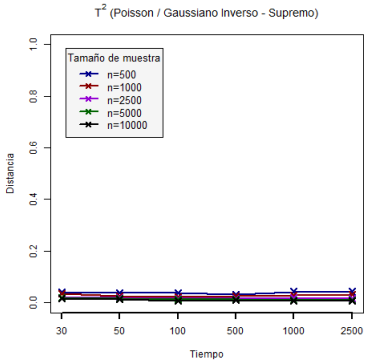
Poisson / Gaussiano Inverso (Hausdorff)						
n \ t	**30**	**50**	**100**	**500**	**1000**	**2500**
500	0.6533	0.5942	0.6126	0.6101	0.6930	0.7311
1000	0.5674	0.4388	0.4853	0.4713	0.4682	0.5529
2500	0.4295	0.3517	0.3777	0.3645	0.3163	0.3021
5000	0.2720	0.3222	0.2807	0.2404	0.2690	0.2491
10000	0.2005	0.2275	0.1798	0.1855	0.1770	0.1997

Tabla B.11: Distancia Hausdorff para perturbaciones a la distribución uniforme en $\mathbb{S}^1$ mediante procesos Poisson $N_1(t)$ e $\mathrm{IG}_2(t)$ de parámetros $\lambda_P = 0.2$ y $\mu_{IG} = 0.2$, $\lambda_{IG} = 0.01$..



Poisson / Gaussiano Inverso (Supremo)						
n t	**30**	**50**	**100**	**500**	**1000**	**2500**
500	0.0393	0.0373	0.0376	0.0324	0.0425	0.0427
1000	0.0355	0.0260	0.0235	0.0258	0.0296	0.0318
2500	0.0191	0.0191	0.0174	0.0152	0.0179	0.0149
5000	0.0195	0.0169	0.0129	0.0130	0.0109	0.0107
10000	0.0159	0.0129	0.0074	0.0092	0.0075	0.0079

Tabla B.12: Distancia del Supremo para perturbaciones a la distribución uniforme en $\mathbb{S}^1$ mediante procesos Poisson $N_1(t)$ e $\mathrm{IG}_2(t)$ de parámetros $\lambda_P = 0.2$ y $\mu_{IG} = 0.2$, $\lambda_{IG} = 0.01$.



Repulsión en hiperplanos (Hausdorff)					
n	**500**	**1000**	**2500**	**5000**	**1000**
Dist.	0.2141	0.1832	0.1564	0.1099	0.1132

Tabla B.13: Distancia Hausdorff Repulsión vs. Uniforme en $\mathbb{S}^1$.

Repulsión en hiperplanos (Supremo)					
n	**500**	**1000**	**2500**	**5000**	**1000**
Dist.	0.1851	0.1695	0.1659	0.1621	0.1563

Tabla B.14: Distancia del Supremo Repulsión vs. Uniforme en $\mathbb{S}^1$.

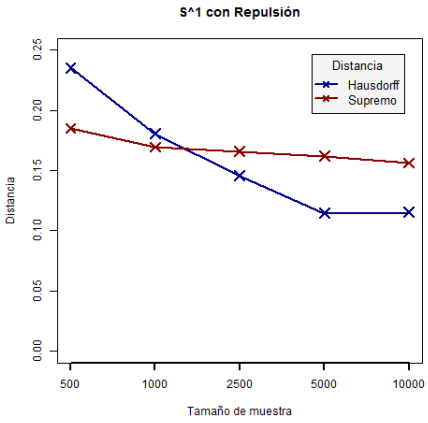


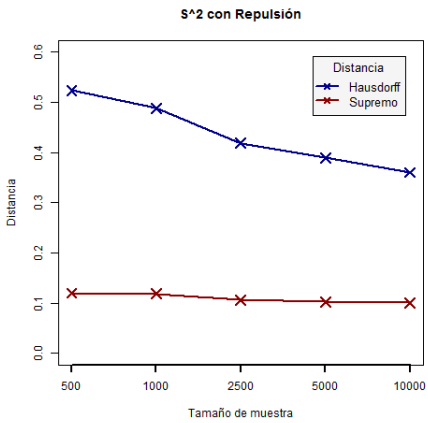
Figura B.1: Prueba

Repulsión en hiperplanos (Hausdorff)					
n	**500**	**1000**	**2500**	**5000**	**1000**
Dist.	0.5240	0.4880	0.4186	0.3894	0.3603

Tabla B.15: Distancia Hausdorff Repulsión vs. Uniforme en $\mathbb{S}^2$.

Repulsión en hiperplanos (Supremo)					
n	**500**	**1000**	**2500**	**5000**	**1000**
Dist.	0.1196	0.1182	0.1058	0.1026	0.1008

Tabla B.16: Distancia del Supremo Repulsión vs. Uniforme en $\mathbb{S}^2$.

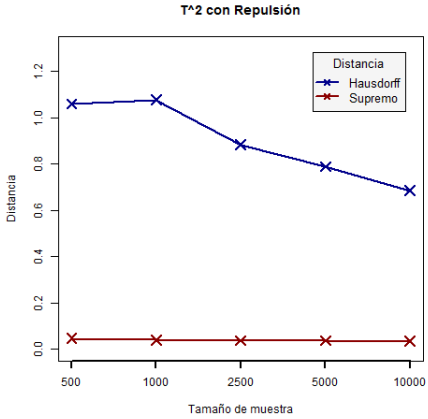


Repulsión en hiperplanos (Hausdorff)					
n	**500**	**1000**	**2500**	**5000**	**1000**
Dist.	1.0595	1.0769	0.8830	0.7887	0.6854

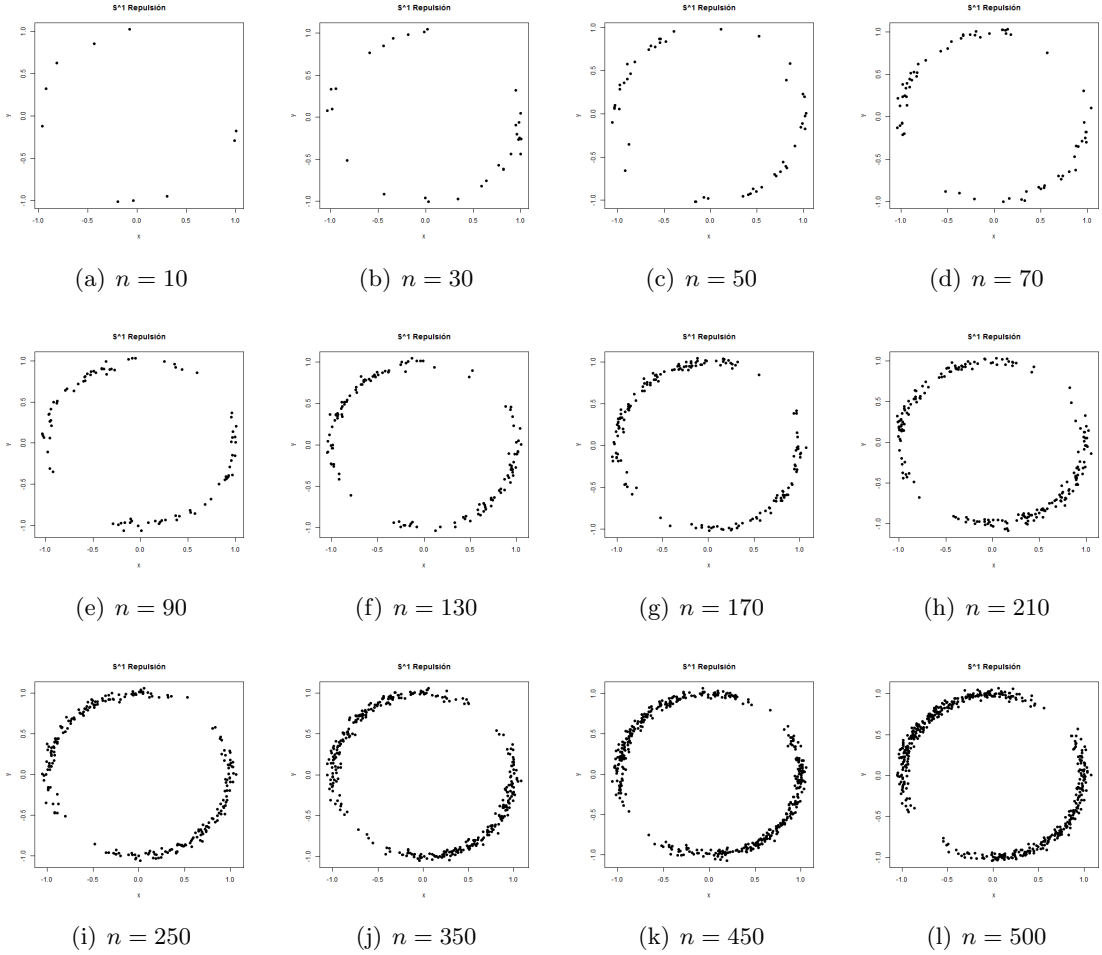
Tabla B.17: Distancia Hausdorff Repulsión vs. Uniforme en $\mathbb{T}^2$.

Repulsión en hiperplanos (Supremo)					
n	**500**	**1000**	**2500**	**5000**	**1000**
Dist.	0.0457	0.0402	0.0375	0.0369	0.0347

Tabla B.18: Distancia del Supremo Repulsión vs. Uniforme en $\mathbb{T}^2$.



Es posible ver en las Tablas B.13 y B.14 que para muestras pequeñas la distancia Hausdorff detecta de mejor manera la diferencia entre la distribución uniforme y la distribución con concentración en hiperplanos sobre $\mathbb{S}^1$. En las parejas de Tablas B.15, B.16 y B.17, B.18 respectivas a $\mathbb{S}^2$ y $\mathbb{T}^2$ vemos cómo la distancia del supremo se mantiene cercana a 0.
A continuación se muestran las gráficas de las nubes de puntos para distintos tamaño de muestra en $\mathbb{S}^1$ con distribución con repulsión en hiperplanos.

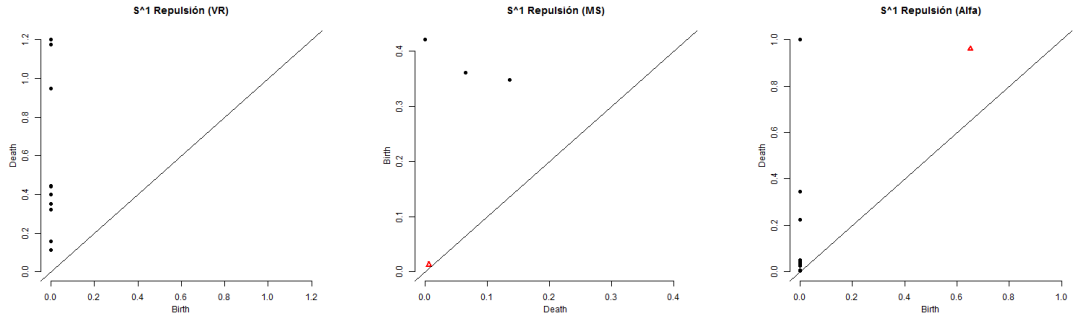


(a) $n = 10$ (b) $n = 30$ (c) $n = 50$ (d) $n = 70$

(e) $n = 90$ (f) $n = 130$ (g) $n = 170$ (h) $n = 210$

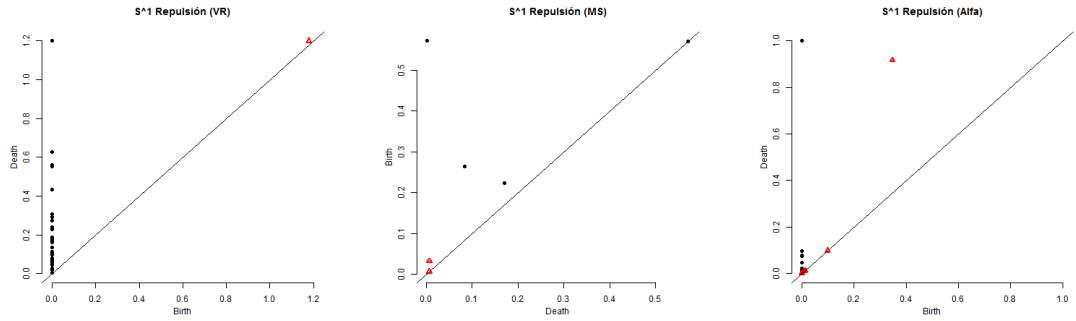
(i) $n = 250$ (j) $n = 350$ (k) $n = 450$ (l) $n = 500$

Figura B.2: Simulación de variables en $\mathbb{S}^1$ con distribución con repulsión en ejes cartesianos para los tamaños de muestra indicados.



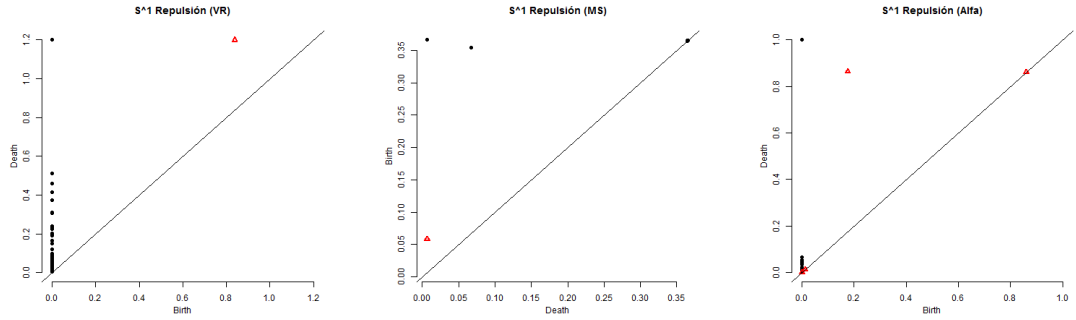
(a) Vietoris-Rips (b) Morse-Smale (c) Alfa

Figura B.3: Diagramas de persistencia para los algoritmos indicados. Simulación de $n = 10$ datos con distribución con repulsión en hiperplanos sobre $\mathbb{S}^1$.



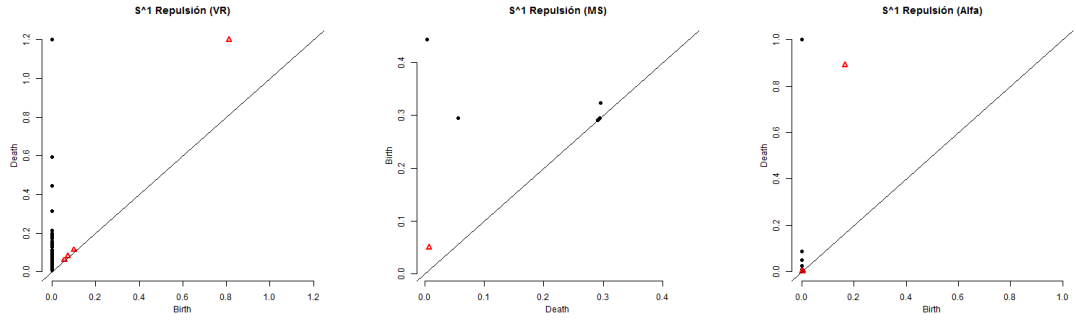
(a) Vietoris-Rips (b) Morse-Smale (c) Alfa

Figura B.4: Diagramas de persistencia para los algoritmos indicados. Simulación de $n = 30$ datos con distribución con repulsión en hiperplanos sobre $\mathbb{S}^1$.



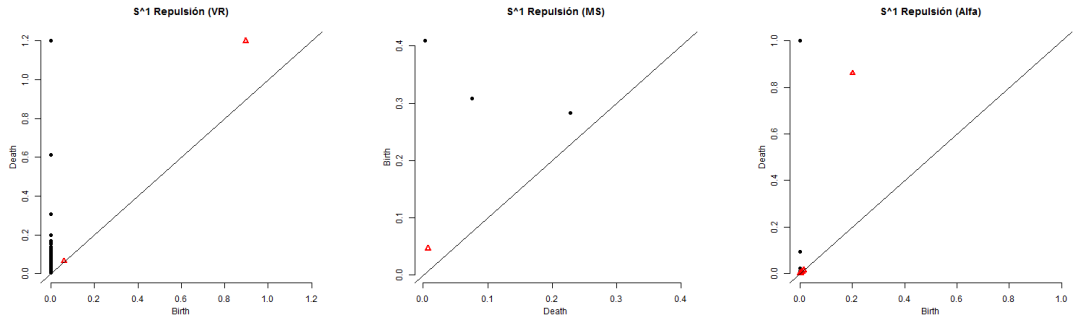
(a) Vietoris-Rips (b) Morse-Smale (c) Alfa

Figura B.5: Diagramas de persistencia para los algoritmos indicados. Simulación de $n = 50$ datos con distribución con repulsión en hiperplanos sobre $\mathbb{S}^1$.



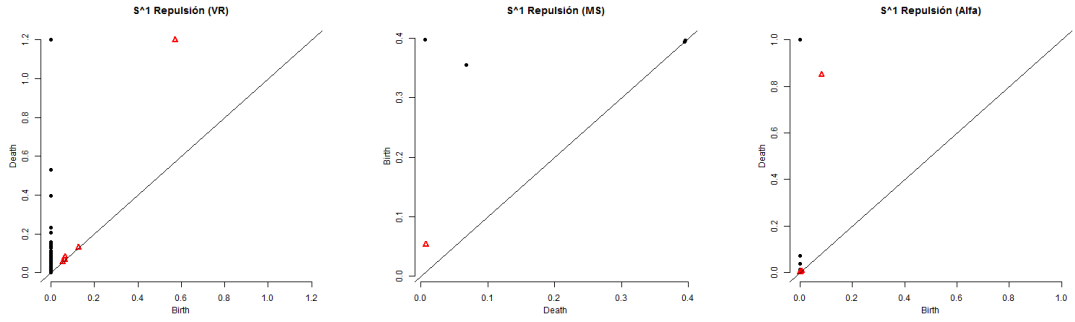
(a) Vietoris-Rips (b) Morse-Smale (c) Alfa

Figura B.6: Diagramas de persistencia para los algoritmos indicados. Simulación de $n = 70$ datos con distribución con repulsión en hiperplanos sobre $\mathbb{S}^1$.



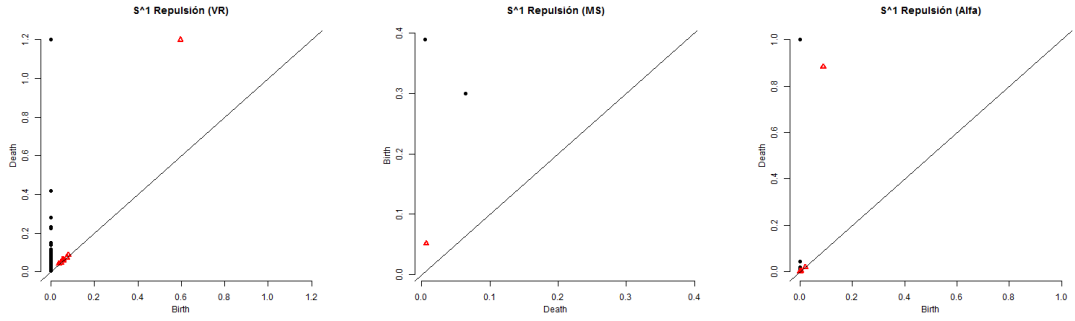
(a) Vietoris-Rips (b) Morse-Smale (c) Alfa

Figura B.7: Diagramas de persistencia para los algoritmos indicados. Simulación de $n = 90$ datos con distribución con repulsión en hiperplanos sobre $\mathbb{S}^1$.



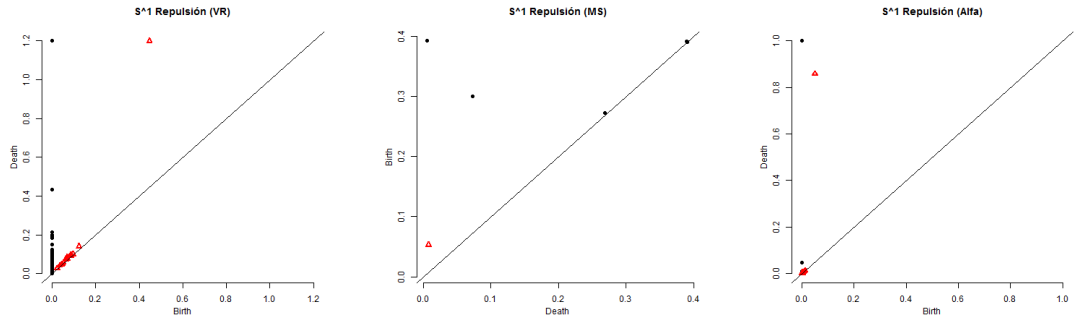
(a) Vietoris-Rips (b) Morse-Smale (c) Alfa

Figura B.8: Diagramas de persistencia para los algoritmos indicados. Simulación de $n = 130$ datos con distribución con repulsión en hiperplanos sobre $\mathbb{S}^1$.



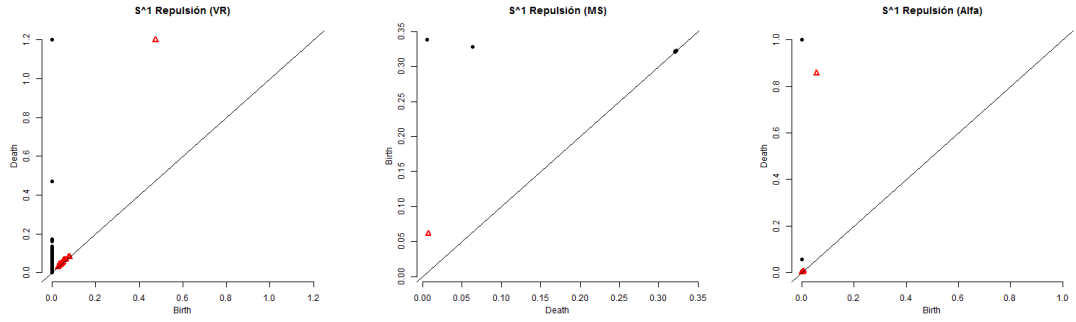
(a) Vietoris-Rips (b) Morse-Smale (c) Alfa

Figura B.9: Diagramas de persistencia para los algoritmos indicados. Simulación de $n = 170$ datos con distribución con repulsión en hiperplanos sobre $\mathbb{S}^1$.



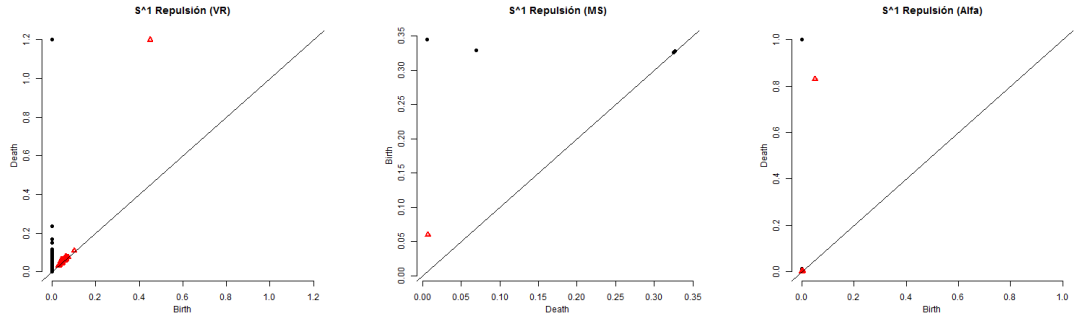
(a) Vietoris-Rips (b) Morse-Smale (c) Alfa

Figura B.10: Diagramas de persistencia para los algoritmos indicados. Simulación de $n = 210$ datos con distribución con repulsión en hiperplanos sobre $\mathbb{S}^1$.



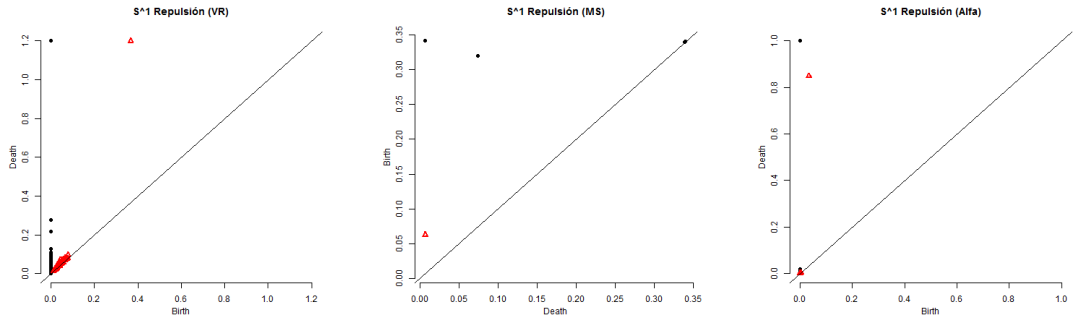
(a) Vietoris-Rips (b) Morse-Smale (c) Alfa

Figura B.11: Diagramas de persistencia para los algoritmos indicados. Simulación de $n = 250$ datos con distribución con repulsión en hiperplanos sobre $\mathbb{S}^1$.



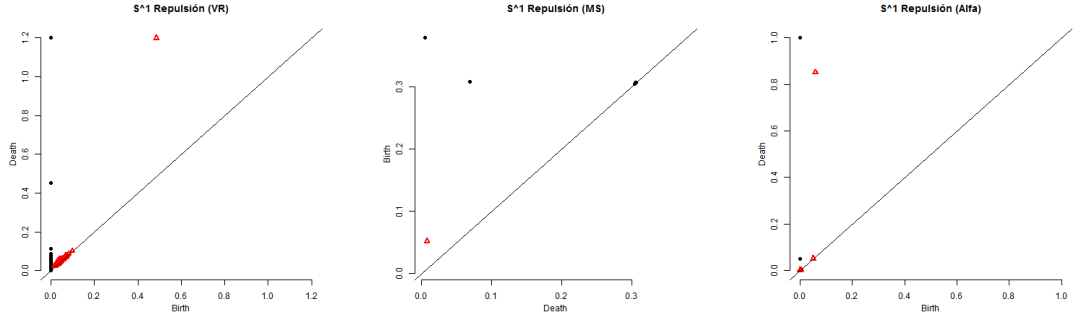
(a) Vietoris-Rips (b) Morse-Smale (c) Alfa

Figura B.12: Diagramas de persistencia para los algoritmos indicados. Simulación de $n = 350$ datos con distribución con repulsión en hiperplanos sobre $\mathbb{S}^1$.



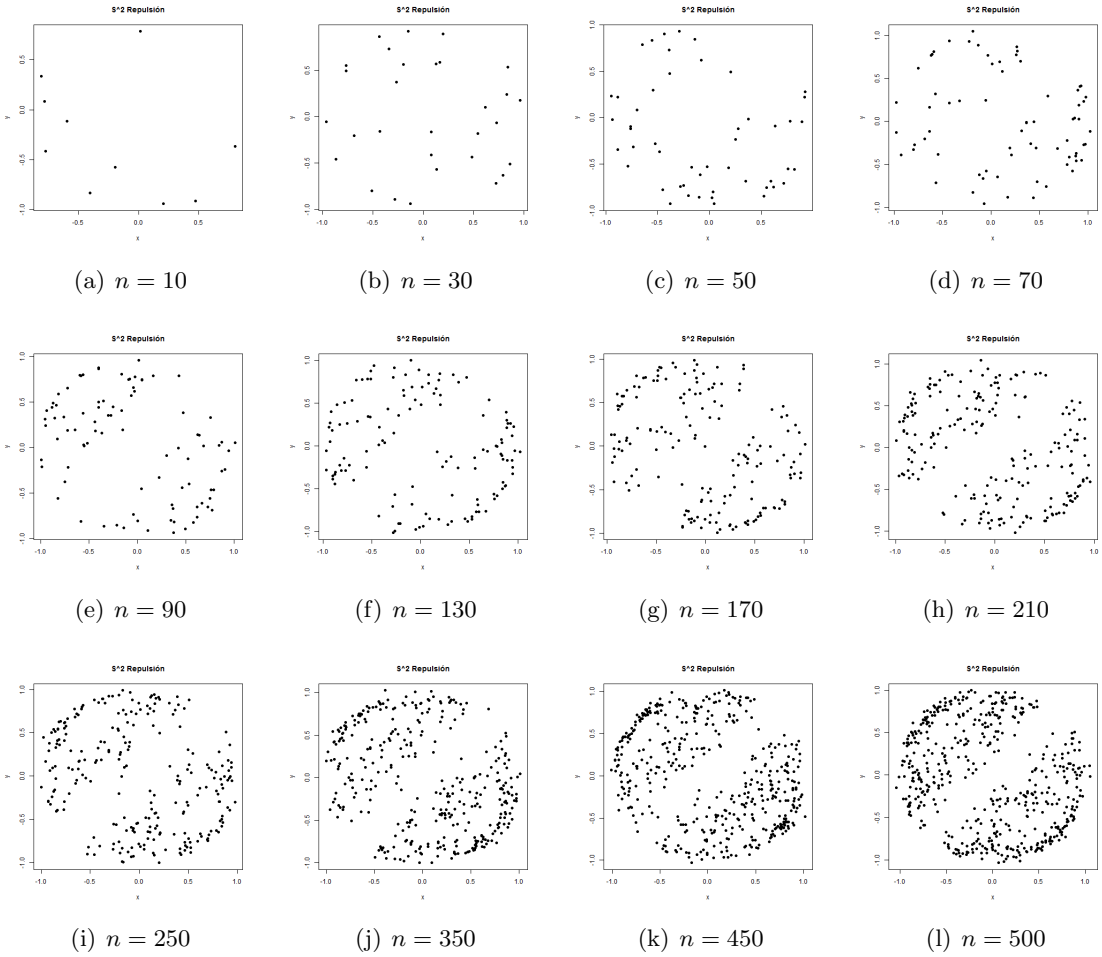
(a) Vietoris-Rips (b) Morse-Smale (c) Alfa

Figura B.13: Diagramas de persistencia para los algoritmos indicados. Simulación de $n = 450$ datos con distribución con repulsión en hiperplanos sobre $\mathbb{S}^1$.



(a) Vietoris-Rips (b) Morse-Smale (c) Alfa

Figura B.14: Diagramas de persistencia para los algoritmos indicados. Simulación de $n = 500$ datos con distribución con repulsión en hiperplanos sobre $\mathbb{S}^1$.

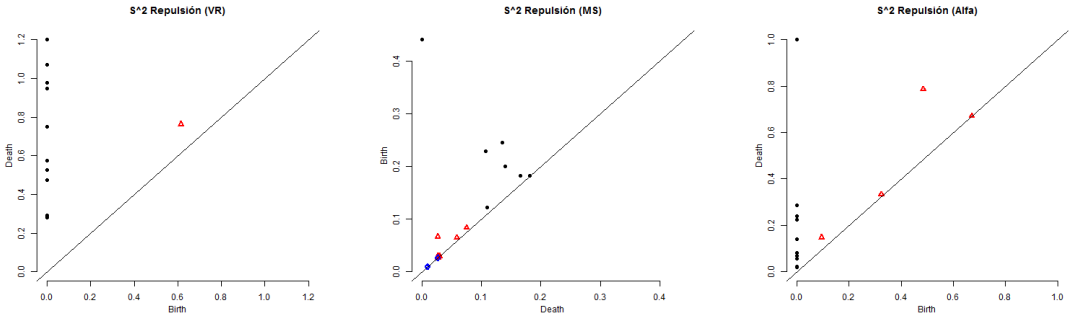


(a) $n = 10$ (b) $n = 30$ (c) $n = 50$ (d) $n = 70$

(e) $n = 90$ (f) $n = 130$ (g) $n = 170$ (h) $n = 210$

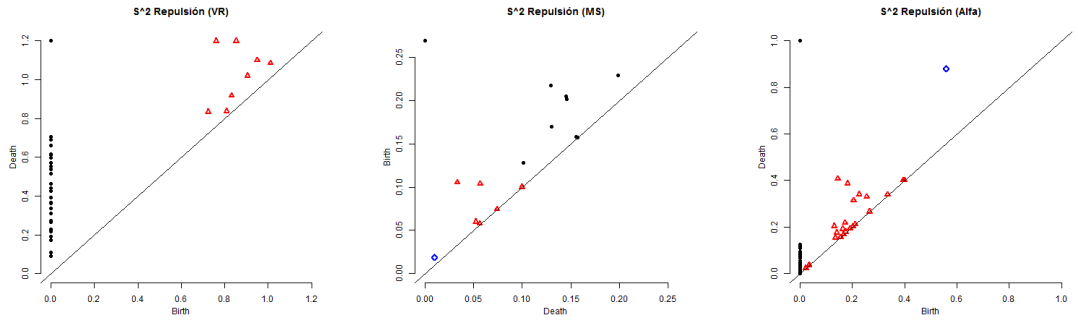
(i) $n = 250$ (j) $n = 350$ (k) $n = 450$ (l) $n = 500$

Figura B.15: Simulación de variables en $\mathbb{S}^2$ con distribución con repulsión en ejes cartesianos para los tamaños de muestra indicados.



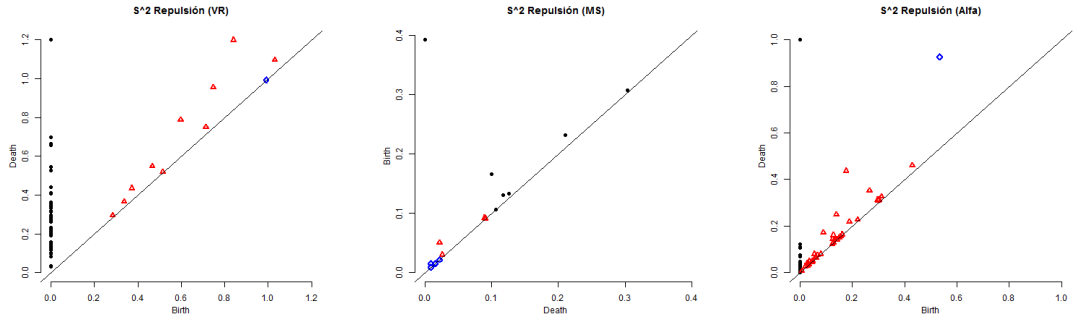
(a) Vietoris-Rips (b) Morse-Smale (c) Alfa

Figura B.16: Diagramas de persistencia para los algoritmos indicados. Simulación de $n = 10$ datos con distribución con repulsión en hiperplanos sobre $\mathbb{S}^2$.



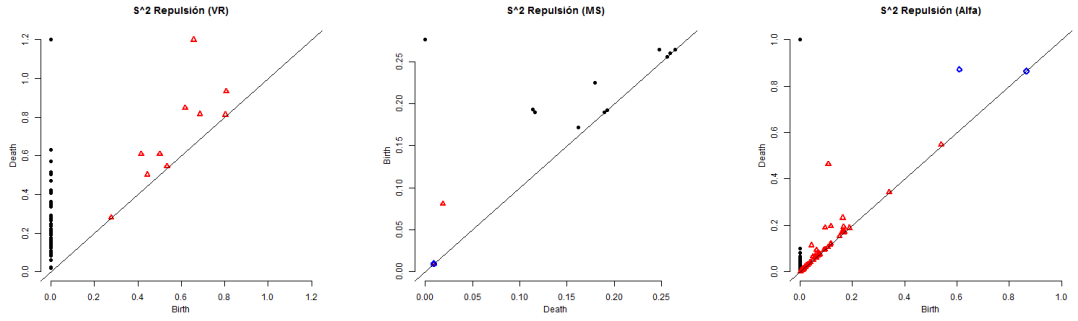
(a) Vietoris-Rips (b) Morse-Smale (c) Alfa

Figura B.17: Diagramas de persistencia para los algoritmos indicados. Simulación de $n = 30$ datos con distribución con repulsión en hiperplanos sobre $\mathbb{S}^2$.



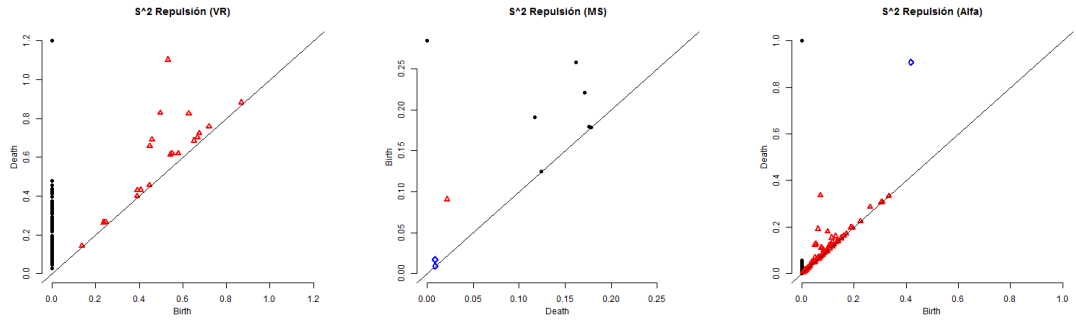
(a) Vietoris-Rips (b) Morse-Smale (c) Alfa

Figura B.18: Diagramas de persistencia para los algoritmos indicados. Simulación de $n = 50$ datos con distribución con repulsión en hiperplanos sobre $\mathbb{S}^2$.



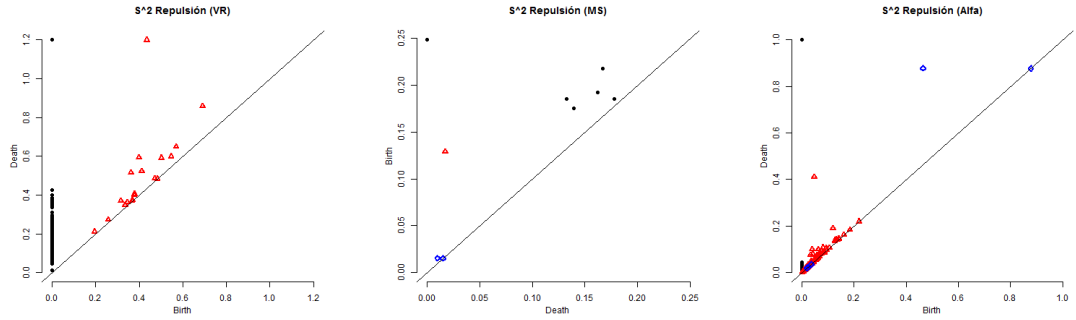
(a) Vietoris-Rips (b) Morse-Smale (c) Alfa

Figura B.19: Diagramas de persistencia para los algoritmos indicados. Simulación de $n = 70$ datos con distribución con repulsión en hiperplanos sobre $\mathbb{S}^2$.



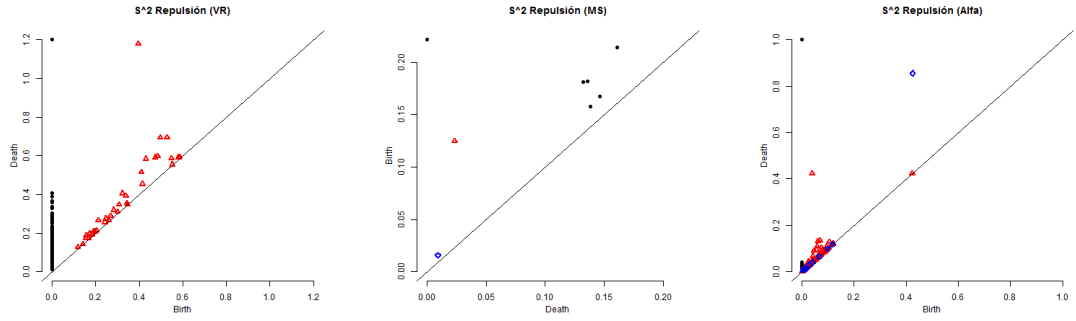
(a) Vietoris-Rips (b) Morse-Smale (c) Alfa

Figura B.20: Diagramas de persistencia para los algoritmos indicados. Simulación de $n = 90$ datos con distribución con repulsión en hiperplanos sobre $\mathbb{S}^2$.



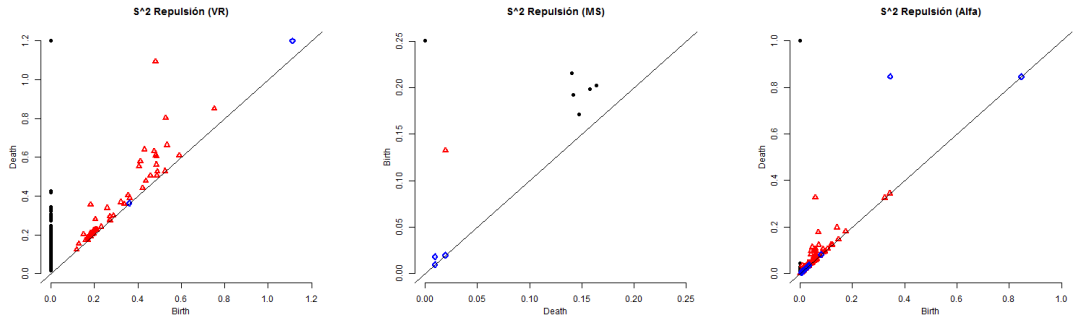
(a) Vietoris-Rips (b) Morse-Smale (c) Alfa

Figura B.21: Diagramas de persistencia para los algoritmos indicados. Simulación de $n = 130$ datos con distribución con repulsión en hiperplanos sobre $\mathbb{S}^2$.



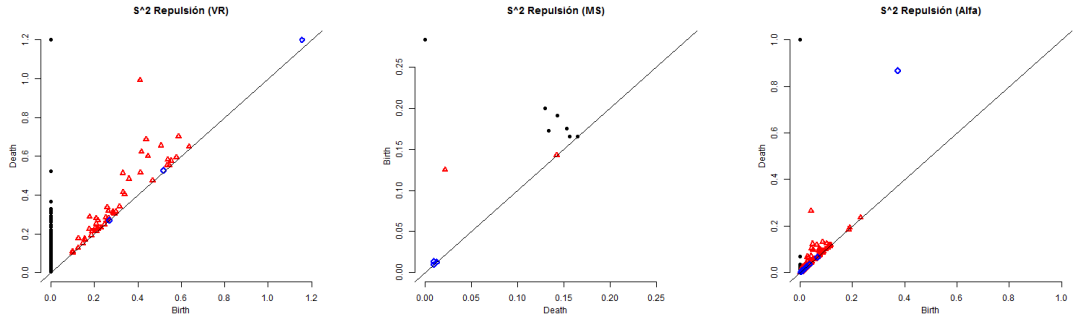
(a) Vietoris-Rips (b) Morse-Smale (c) Alfa

Figura B.22: Diagramas de persistencia para los algoritmos indicados. Simulación de $n = 170$ datos con distribución con repulsión en hiperplanos sobre $\mathbb{S}^2$.



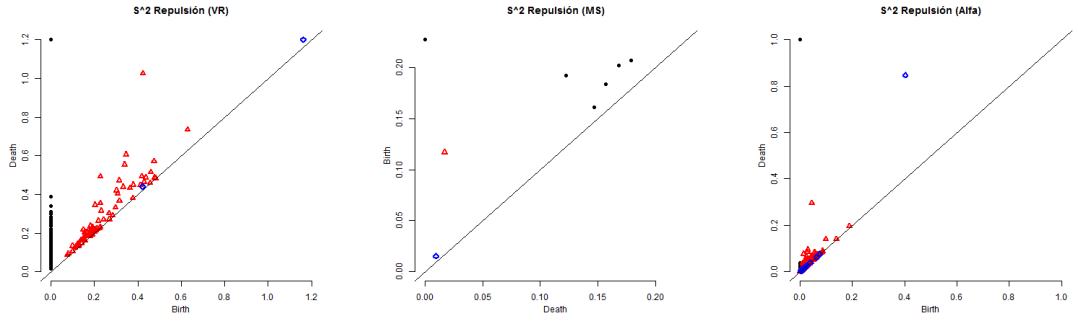
(a) Vietoris-Rips (b) Morse-Smale (c) Alfa

Figura B.23: Diagramas de persistencia para los algoritmos indicados. Simulación de $n = 210$ datos con distribución con repulsión en hiperplanos sobre $\mathbb{S}^2$.



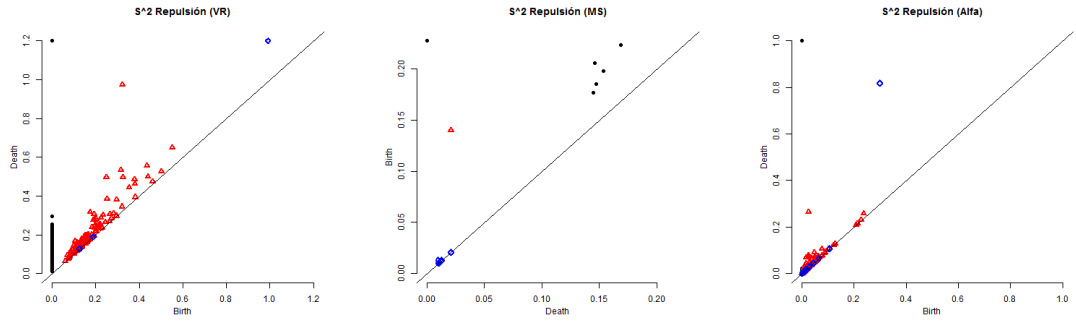
(a) Vietoris-Rips (b) Morse-Smale (c) Alfa

Figura B.24: Diagramas de persistencia para los algoritmos indicados. Simulación de $n = 250$ datos con distribución con repulsión en hiperplanos sobre $\mathbb{S}^2$.



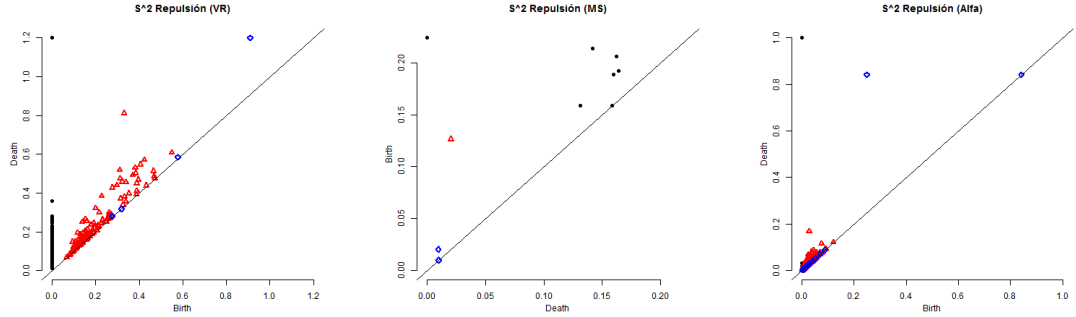
(a) Vietoris-Rips (b) Morse-Smale (c) Alfa

Figura B.25: Diagramas de persistencia para los algoritmos indicados. Simulación de $n = 350$ datos con distribución con repulsión en hiperplanos sobre $\mathbb{S}^2$.



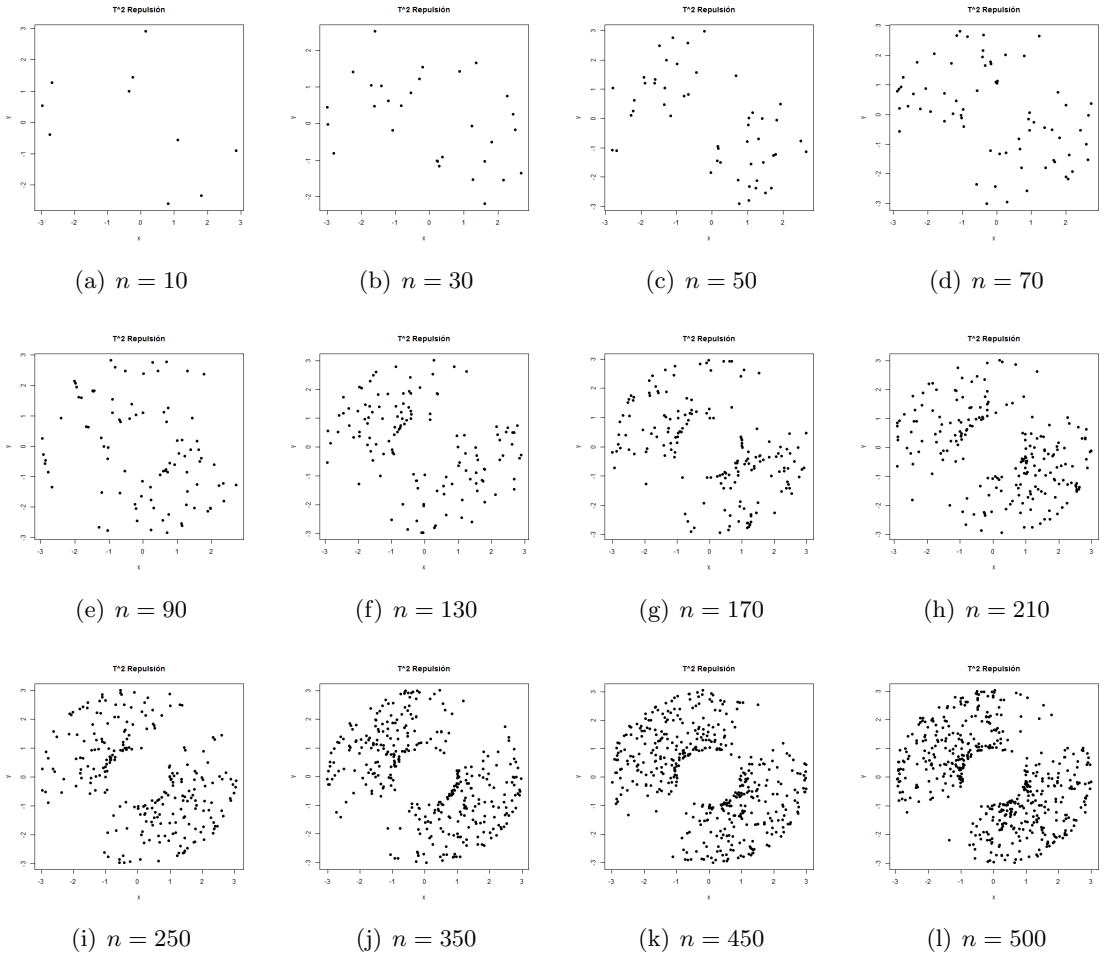
(a) Vietoris-Rips (b) Morse-Smale (c) Alfa

Figura B.26: Diagramas de persistencia para los algoritmos indicados. Simulación de $n = 450$ datos con distribución con repulsión en hiperplanos sobre $\mathbb{S}^2$.



(a) Vietoris-Rips (b) Morse-Smale (c) Alfa

Figura B.27: Diagramas de persistencia para los algoritmos indicados. Simulación de $n = 500$ datos con distribución con repulsión en hiperplanos sobre $\mathbb{S}^2$.

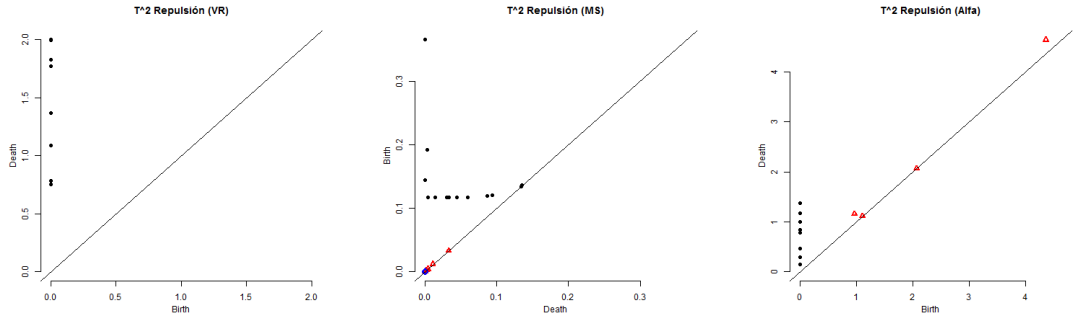


(a) $n = 10$ (b) $n = 30$ (c) $n = 50$ (d) $n = 70$

(e) $n = 90$ (f) $n = 130$ (g) $n = 170$ (h) $n = 210$

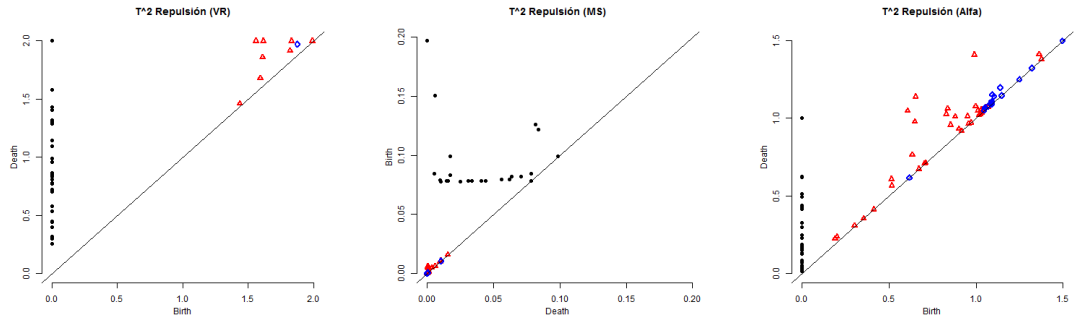
(i) $n = 250$ (j) $n = 350$ (k) $n = 450$ (l) $n = 500$

Figura B.28: Simulación de variables en $\mathbb{T}^2$ con distribución con repulsión en ejes cartesianos para los tamaños de muestra indicados.



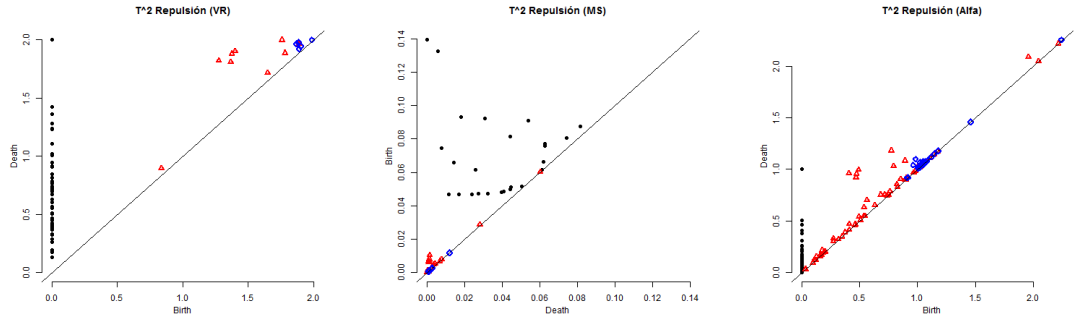
(a) Vietoris-Rips (b) Morse-Smale (c) Alfa

Figura B.29: Diagramas de persistencia para los algoritmos indicados. Simulación de $n = 10$ datos con distribución con repulsión en hiperplanos sobre $\mathbb{T}^2$.



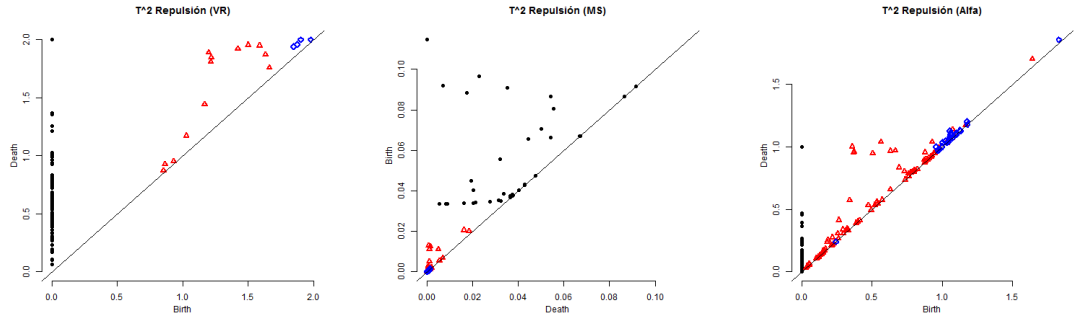
(a) Vietoris-Rips (b) Morse-Smale (c) Alfa

Figura B.30: Diagramas de persistencia para los algoritmos indicados. Simulación de $n = 30$ datos con distribución con repulsión en hiperplanos sobre $\mathbb{T}^2$.



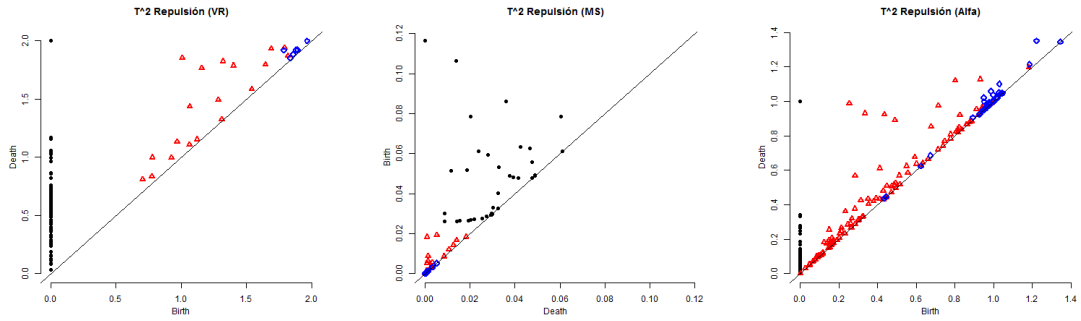
(a) Vietoris-Rips (b) Morse-Smale (c) Alfa

Figura B.31: Diagramas de persistencia para los algoritmos indicados. Simulación de $n = 50$ datos con distribución con repulsión en hiperplanos sobre $\mathbb{T}^2$.



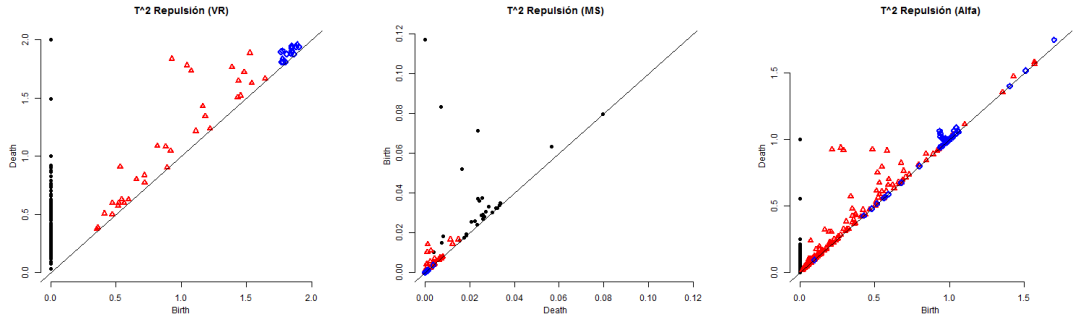
(a) Vietoris-Rips (b) Morse-Smale (c) Alfa

Figura B.32: Diagramas de persistencia para los algoritmos indicados. Simulación de $n = 70$ datos con distribución con repulsión en hiperplanos sobre $\mathbb{T}^2$.



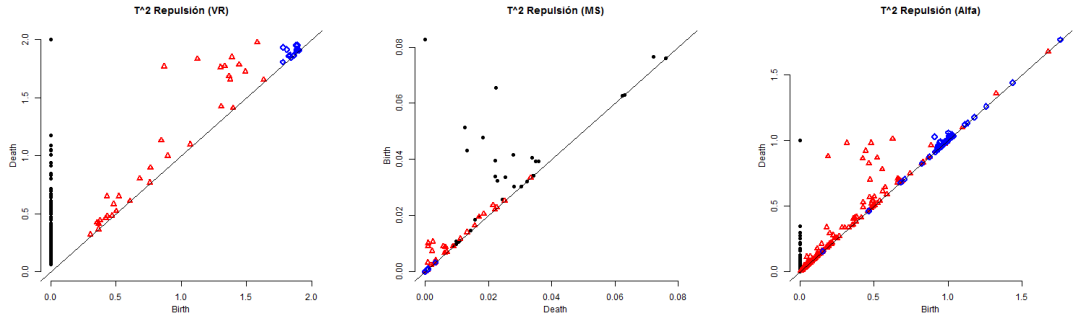
(a) Vietoris-Rips (b) Morse-Smale (c) Alfa

Figura B.33: Diagramas de persistencia para los algoritmos indicados. Simulación de $n = 90$ datos con distribución con repulsión en hiperplanos sobre $\mathbb{T}^2$.



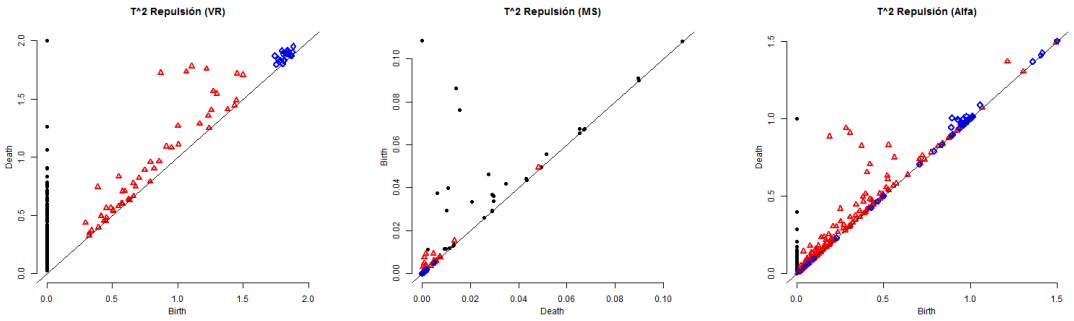
(a) Vietoris-Rips (b) Morse-Smale (c) Alfa

Figura B.34: Diagramas de persistencia para los algoritmos indicados. Simulación de $n = 130$ datos con distribución con repulsión en hiperplanos sobre $\mathbb{T}^2$.



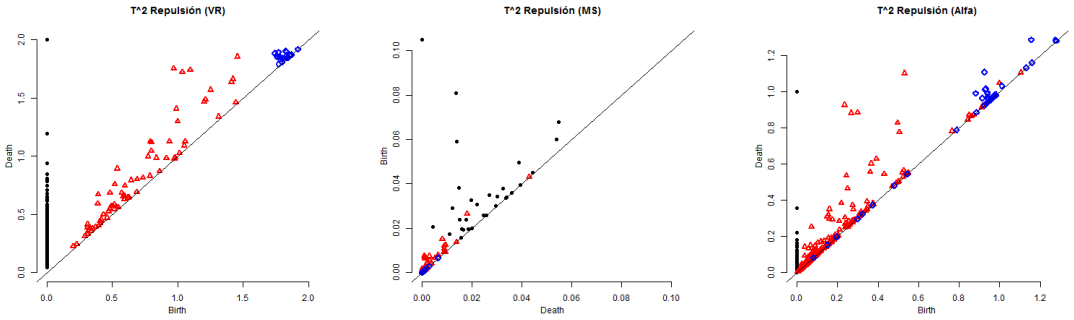
(a) Vietoris-Rips (b) Morse-Smale (c) Alfa

Figura B.35: Diagramas de persistencia para los algoritmos indicados. Simulación de $n = 170$ datos con distribución con repulsión en hiperplanos sobre $\mathbb{T}^2$.



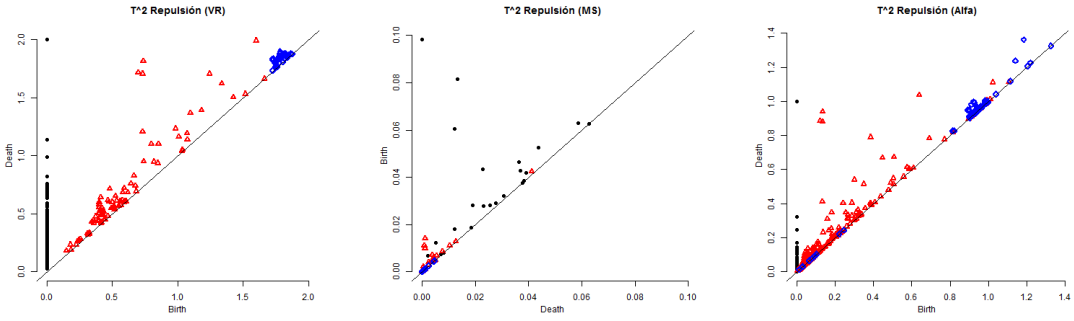
(a) Vietoris-Rips (b) Morse-Smale (c) Alfa

Figura B.36: Diagramas de persistencia para los algoritmos indicados. Simulación de $n = 210$ datos con distribución con repulsión en hiperplanos sobre $\mathbb{T}^2$.



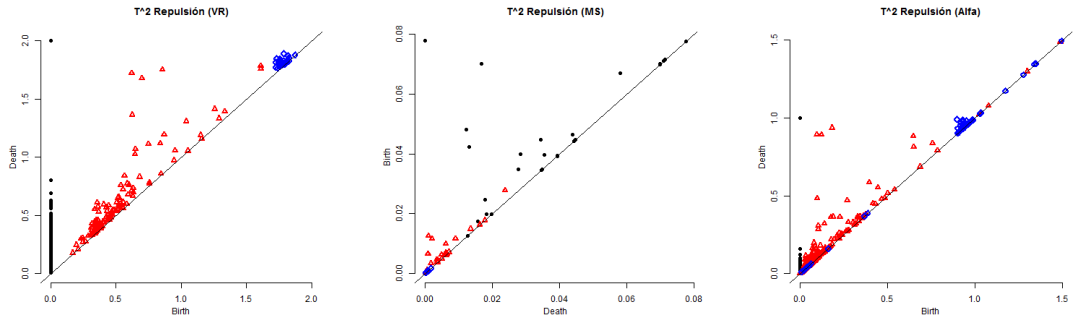
(a) Vietoris-Rips (b) Morse-Smale (c) Alfa

Figura B.37: Diagramas de persistencia para los algoritmos indicados. Simulación de $n = 250$ datos con distribución con repulsión en hiperplanos sobre $\mathbb{T}^2$.



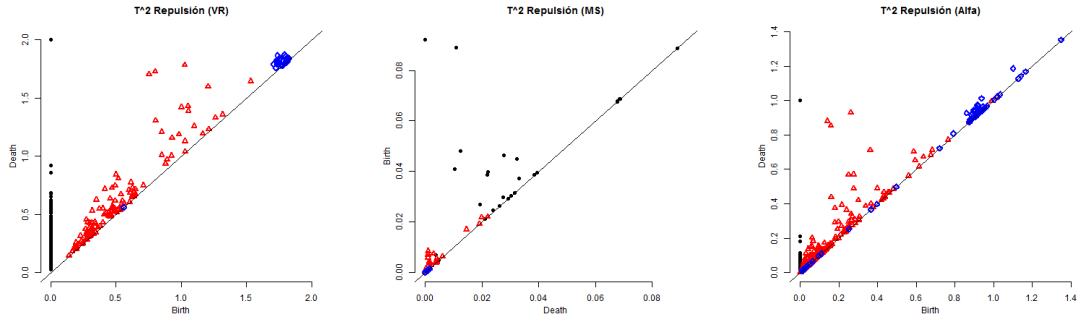
(a) Vietoris-Rips (b) Morse-Smale (c) Alfa

Figura B.38: Diagramas de persistencia para los algoritmos indicados. Simulación de $n = 350$ datos con distribución con repulsión en hiperplanos sobre $\mathbb{T}^2$.



(a) Vietoris-Rips (b) Morse-Smale (c) Alfa

Figura B.39: Diagramas de persistencia para los algoritmos indicados. Simulación de $n = 450$ datos con distribución con repulsión en hiperplanos sobre $\mathbb{T}^2$.



(a) Vietoris-Rips (b) Morse-Smale (c) Alfa

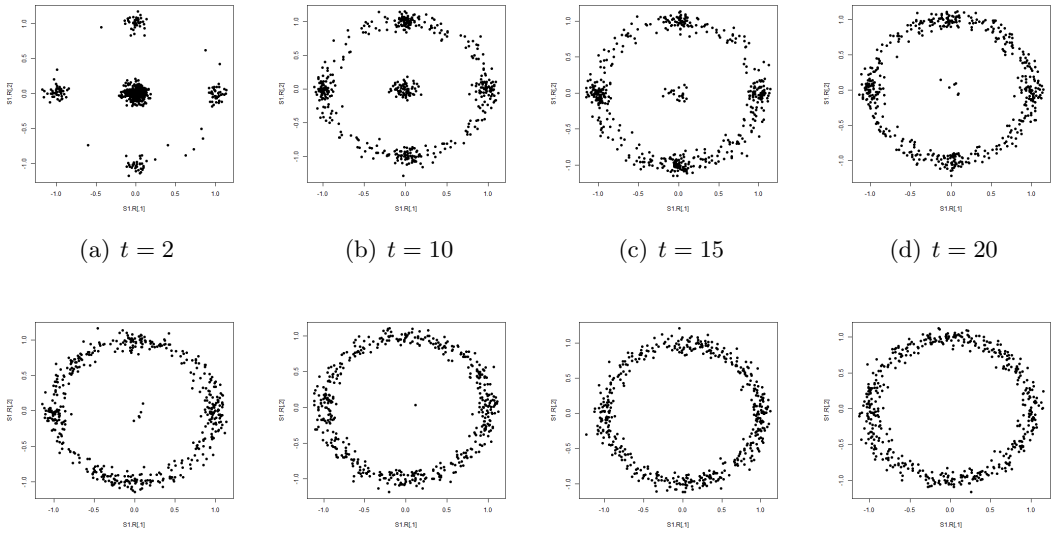
Figura B.40: Diagramas de persistencia para los algoritmos indicados. Simulación de $n = 500$ datos con distribución con repulsión en hiperplanos sobre $\mathbb{T}^2$.

C

Perturbaciones a distribuciones en $\mathbb{S}^1, \mathbb{S}^2$ y $\mathbb{T}^2$

En este apéndice se muestran una serie de ejemplos de perturbaciones a la distribución uniforme sobre las variedades $\mathbb{S}^1$, $\mathbb{S}^2$ y $\mathbb{T}^2$. Se utilizan los algoritmos del Apéndice A, así como la teoría presentada en el Capítulo 1. El objetivo es ilustrar la convergencia a la distribución uniforme en estas variedades dada por la Proposición 3(a). Para esto se realizan distintas combinaciones de procesos con medias y varianzas iguales. También se simula con medias iguales y varianzas distintas. Otro caso es medias distintas y varianzas iguales. En cada figura indicamos los valores de estos parámetros para que el lector pueda entender el efecto de estas combinaciones.

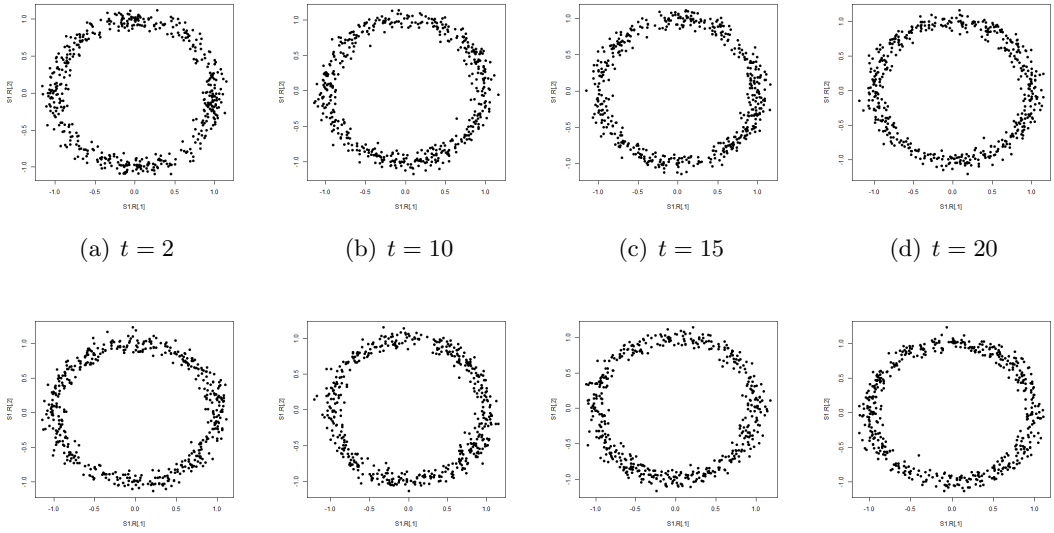
C.1. Perturbaciones mediante procesos Poisson (PP)



(a) $t = 2$ (b) $t = 10$ (c) $t = 15$ (d) $t = 20$

(e) $t = 25$ (f) $t = 30$ (g) $t = 40$ (h) $t = 50$

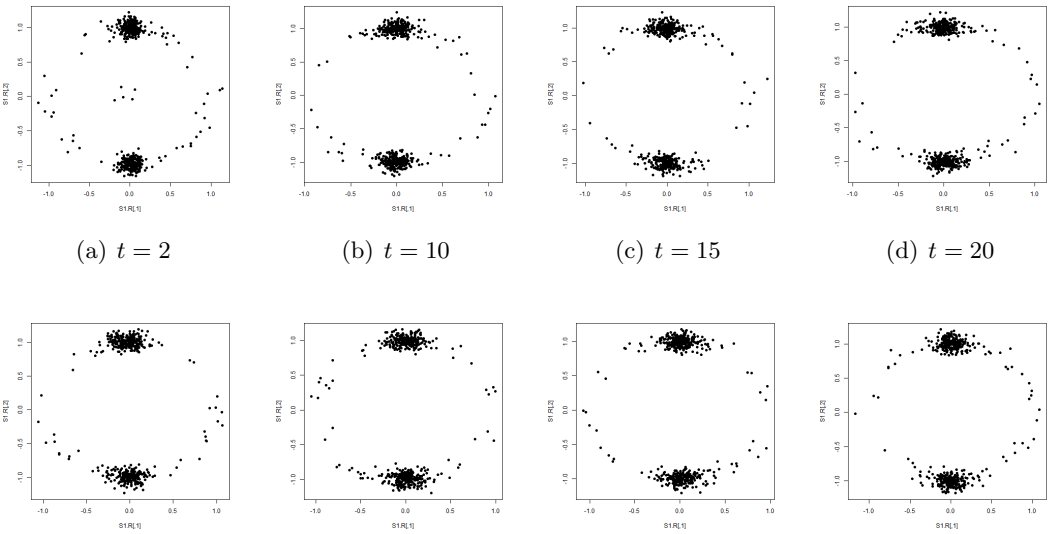
Figura C.1: 500 datos sobre $\mathbb{S}^1$ con coeficientes PP $N_1(t)$ y $N_2(t)$ de parámetros $\lambda_1 = \lambda_2 = 0.1$.



(a) $t = 2$ (b) $t = 10$ (c) $t = 15$ (d) $t = 20$

(e) $t = 25$ (f) $t = 30$ (g) $t = 40$ (h) $t = 50$

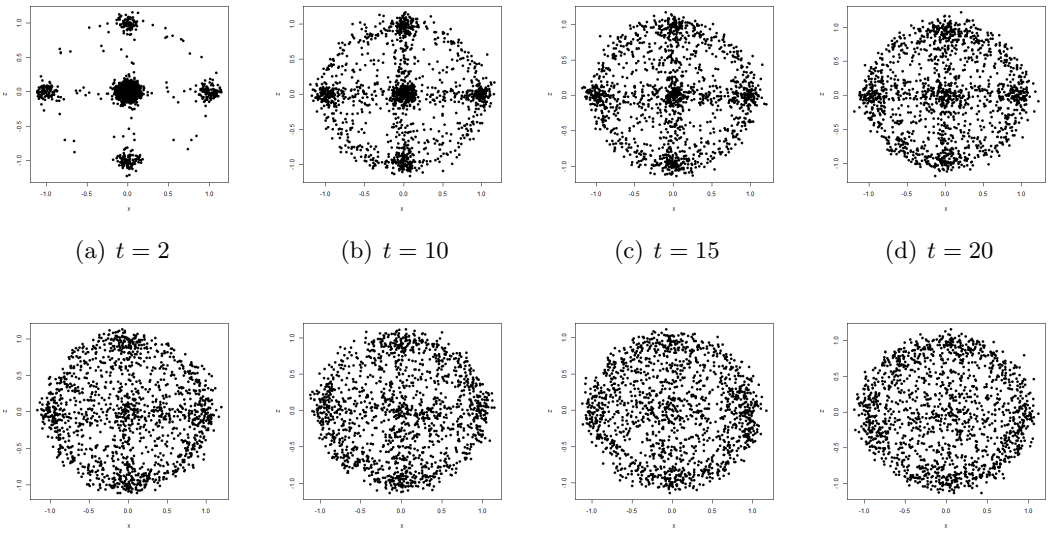
Figura C.2: 500 datos sobre $\mathbb{S}^1$ con coeficientes PP $N_1(t)$ y $N_2(t)$ de parámetros $\lambda_1 = \lambda_2 = 2$.



(a) $t = 2$ (b) $t = 10$ (c) $t = 15$ (d) $t = 20$

(e) $t = 25$ (f) $t = 30$ (g) $t = 40$ (h) $t = 50$

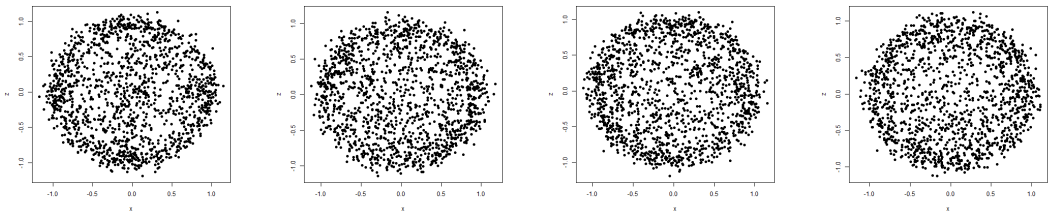
Figura C.3: 500 datos sobre $\mathbb{S}^1$ con coeficientes PP $N_1(t)$ y $N_2(t)$ de parámetros $\lambda_1 = 0.1$ y $\lambda_2 = 2$ respectivamente.



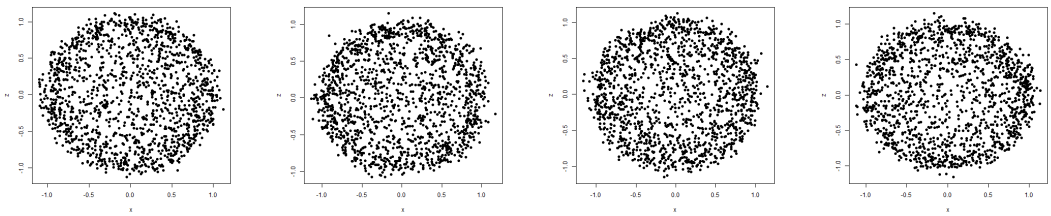
(a) $t = 2$ (b) $t = 10$ (c) $t = 15$ (d) $t = 20$

(e) $t = 25$ (f) $t = 30$ (g) $t = 40$ (h) $t = 50$

Figura C.4: 1500 datos sobre $\mathbb{S}^2$ con coeficientes PP $N_1(t)$ y $N_2(t)$ de parámetros $\lambda_1 = \lambda_2 = 0.1$.

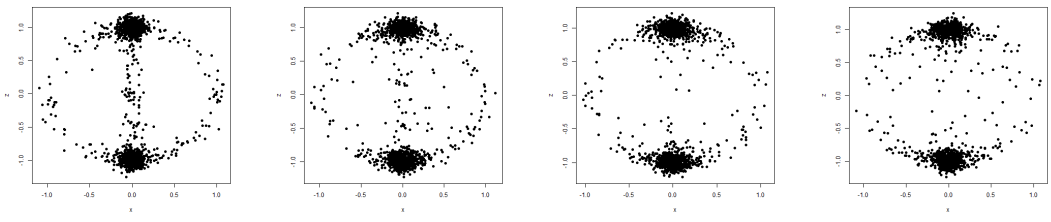


(a) $t = 2$ (b) $t = 10$ (c) $t = 15$ (d) $t = 20$

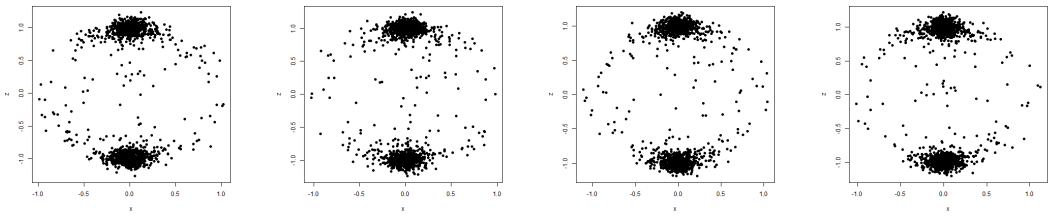


(e) $t = 25$ (f) $t = 30$ (g) $t = 40$ (h) $t = 50$

Figura C.5: 1500 datos sobre $\mathbb{S}^2$ con coeficientes PP $N_1(t)$ y $N_2(t)$ de parámetro $\lambda_1 = \lambda_2 = 2$.

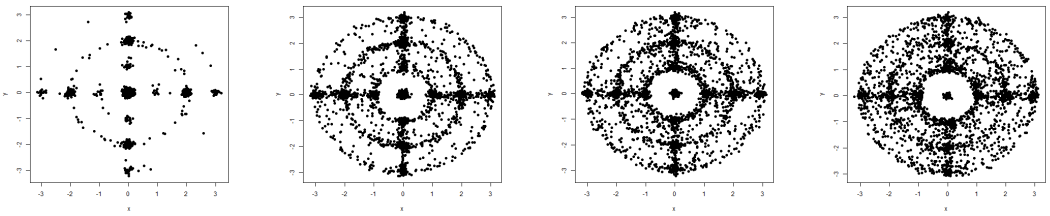


(a) $t = 2$ (b) $t = 10$ (c) $t = 15$ (d) $t = 20$

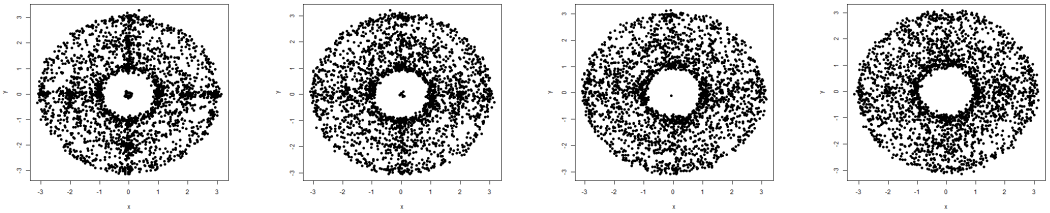


(e) $t = 25$ (f) $t = 30$ (g) $t = 40$ (h) $t = 50$

Figura C.6: 1500 datos sobre $\mathbb{S}^2$ con coeficientes PP $N_1(t)$ y $N_2(t)$ de parámetros $\lambda_1 = 0.1$ y $\lambda_2 = 2$ respectivamente.

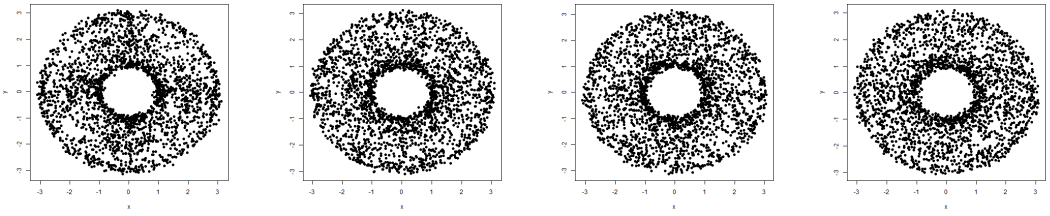


(a) $t = 2$ (b) $t = 10$ (c) $t = 15$ (d) $t = 20$

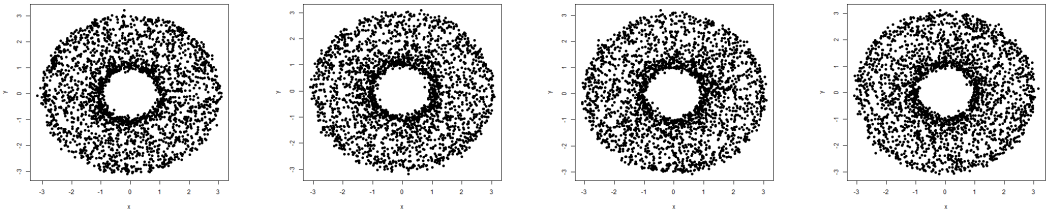


(e) $t = 25$ (f) $t = 30$ (g) $t = 40$ (h) $t = 50$

Figura C.7: 2500 datos sobre $\mathbb{T}^2$ con coeficientes PP $N_1(t)$ y $N_2(t)$ de parámetros $\lambda_1 = \lambda_2 = 0.1$.

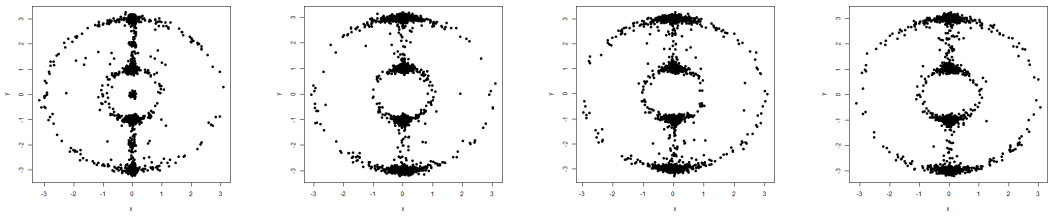


(a) $t = 2$ (b) $t = 10$ (c) $t = 15$ (d) $t = 20$

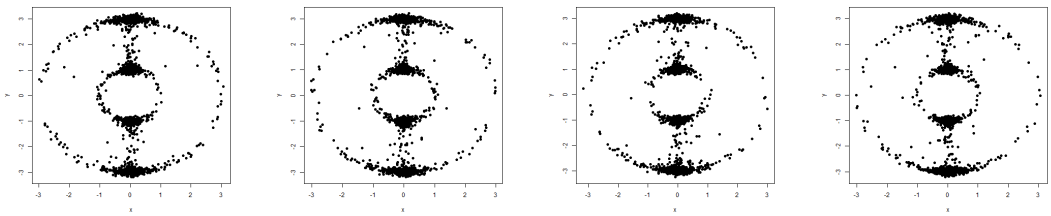


(e) $t = 25$ (f) $t = 30$ (g) $t = 40$ (h) $t = 50$

Figura C.8: 2500 datos sobre $\mathbb{T}^2$ con coeficientes PP $N_1(t)$ y $N_2(t)$ de parámetro $\lambda_1 = \lambda_2 = 2$.



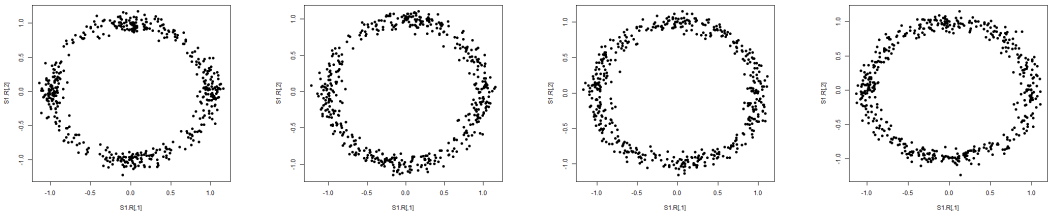
(a) $t = 2$ (b) $t = 10$ (c) $t = 15$ (d) $t = 20$



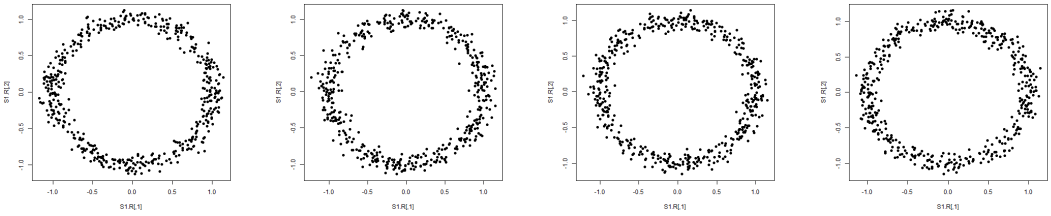
(e) $t = 25$ (f) $t = 30$ (g) $t = 40$ (h) $t = 50$

Figura C.9: 2500 datos sobre $\mathbb{T}^2$ con coeficientes PP $N_1(t)$ y $N_2(t)$ de parámetros $\lambda_1 = 0.1$ y $\lambda_2 = 2$ respectivamente.

C.2. Perturbaciones mediante procesos Inverso Gaussiano

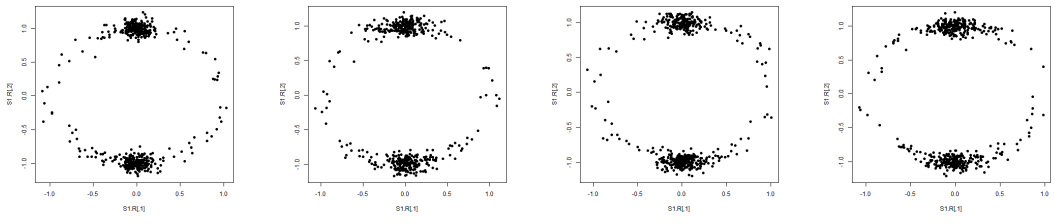


(a) $t = 2$ (b) $t = 10$ (c) $t = 15$ (d) $t = 20$

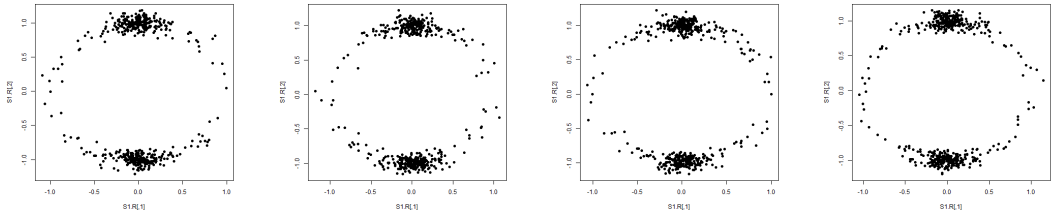


(e) $t = 25$ (f) $t = 30$ (g) $t = 40$ (h) $t = 50$

Figura C.10: 500 datos sobre $\mathbb{S}^1$ con coeficientes IG $I_1(t)$ y $I_2(t)$ con $\mathbb{E}(I_1(t)) = \mathbb{E}(I_2(t)) = 0.1$ y $\mathrm{Var}(I_1(t)) = \mathrm{Var}(I_2(t)) = 0.1$.

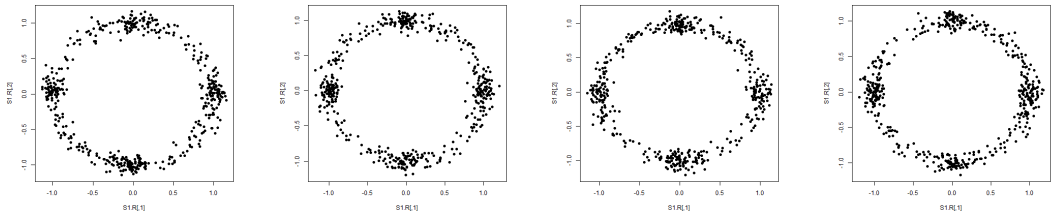


(a) $t = 2$ (b) $t = 10$ (c) $t = 15$ (d) $t = 20$

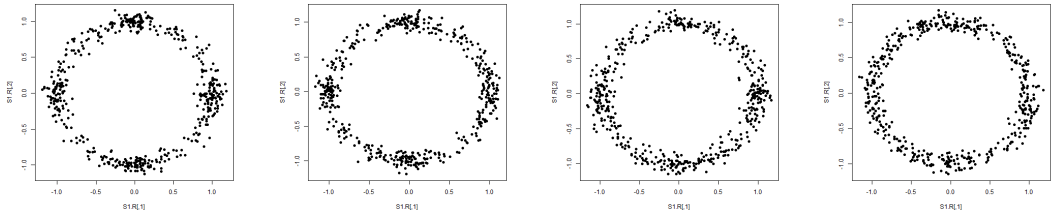


(e) $t = 25$ (f) $t = 30$ (g) $t = 40$ (h) $t = 50$

Figura C.11: 500 datos sobre $\mathbb{S}^1$ con coeficientes IG $I_1(t)$ y $I_2(t)$ con $\mathbb{E}(I_1(t)) = 0.1$, $\mathbb{E}(I_2(t)) = 1$ y $\mathrm{Var}(I_1(t)) = \mathrm{Var}(I_2(t)) = 0.1$.

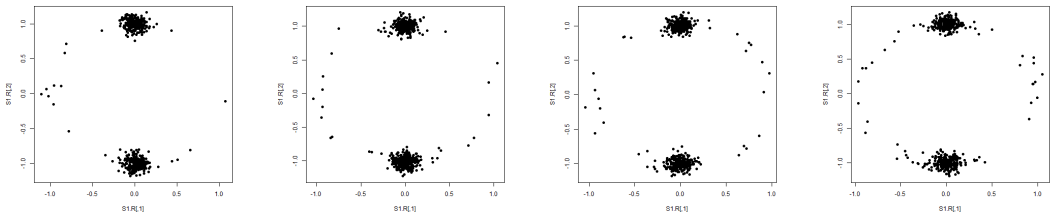


(a) $t = 2$ (b) $t = 10$ (c) $t = 25$ (d) $t = 50$

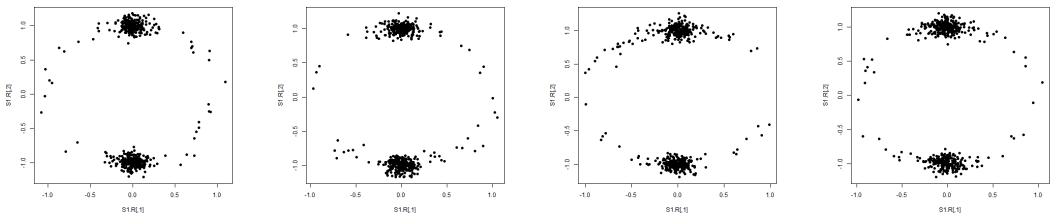


(e) $t = 100$ (f) $t = 300$ (g) $t = 500$ (h) $t = 1000$

Figura C.12: 500 datos sobre $\mathbb{S}^1$ con coeficientes IG $I_1(t)$ y $I_2(t)$ con $\mathbb{E}(I_1(t)) = \mathbb{E}(I_2(t)) = 0.1$ y $\mathrm{Var}(I_1(t)) = \mathrm{Var}(I_2(t)) = 10$.

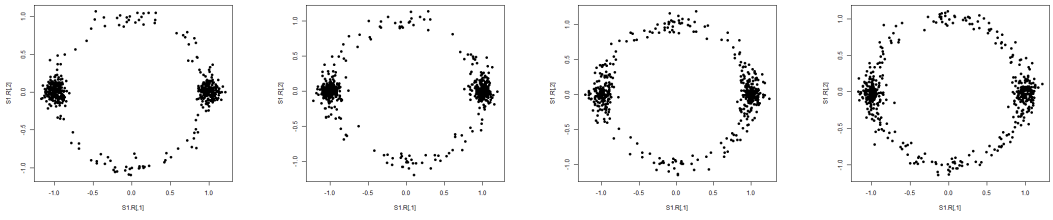


(a) $t = 2$ (b) $t = 10$ (c) $t = 25$ (d) $t = 50$

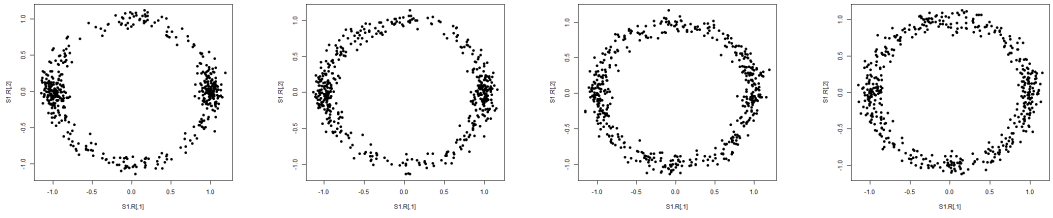


(e) $t = 100$ (f) $t = 300$ (g) $t = 500$ (h) $t = 1000$

Figura C.13: 500 datos sobre $\mathbb{S}^1$ con coeficientes IG $I_1(t)$ y $I_2(t)$ con $\mathbb{E}(I_1(t)) = 0.1, = \mathbb{E}(I_2(t)) = 2$ y $\mathrm{Var}(I_1(t)) = \mathrm{Var}(I_2(t)) = 10$.

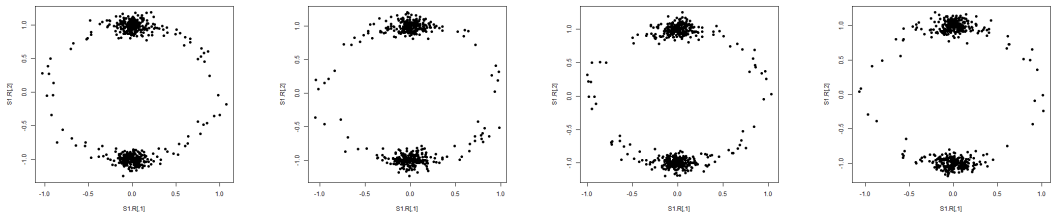


(a) $t = 2$ (b) $t = 10$ (c) $t = 25$ (d) $t = 50$

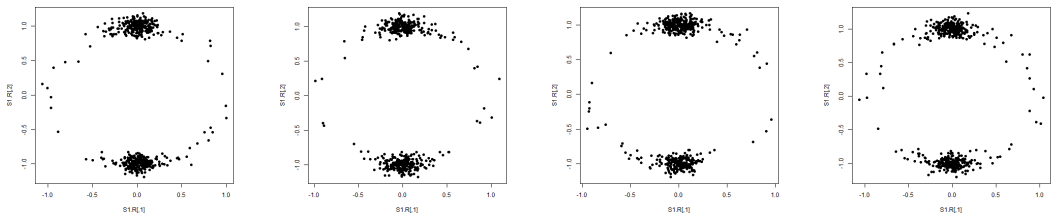


(e) $t = 100$ (f) $t = 300$ (g) $t = 500$ (h) $t = 1000$

Figura C.14: 500 datos sobre $\mathbb{S}^1$ con coeficientes IG $I_1(t)$ y $I_2(t)$ con $\mathbb{E}(I_1(t)) = \mathbb{E}(I_2(t)) = 0.1$ y $\mathrm{Var}(I_1(t)) = 1, \mathrm{Var}(I_2(t)) = 10$.

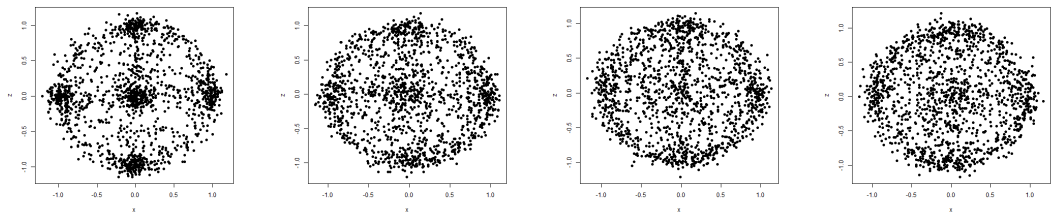


(a) $t = 2$ (b) $t = 10$ (c) $t = 25$ (d) $t = 50$

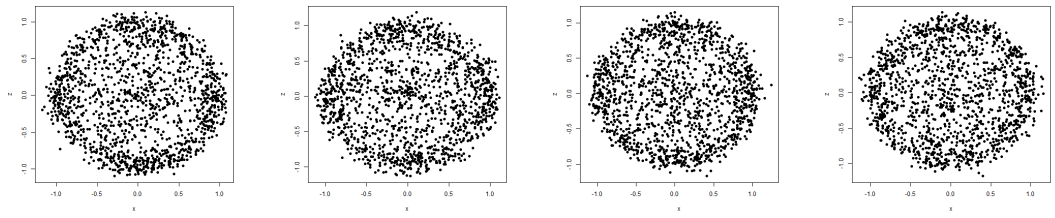


(e) $t = 100$ (f) $t = 300$ (g) $t = 500$ (h) $t = 1000$

Figura C.15: 500 datos sobre $\mathbb{S}^1$ con coeficientes IG $I_1(t)$ y $I_2(t)$ con $\mathbb{E}(I_1(t)) = 0.1, = \mathbb{E}(I_2(t)) = 2$ y $\mathrm{Var}(I_1(t)) = 0.1, = \mathrm{Var}(I_2(t)) = 10$.

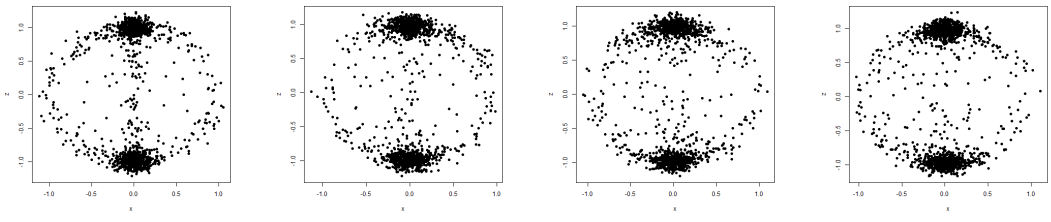


(a) $t = 2$ (b) $t = 10$ (c) $t = 15$ (d) $t = 20$

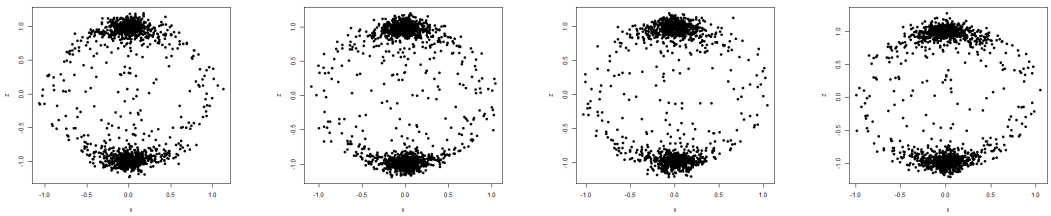


(e) $t = 25$ (f) $t = 30$ (g) $t = 40$ (h) $t = 50$

Figura C.16: 1500 datos sobre $\mathbb{S}^2$ con coeficientes IG $I_1(t)$,$I_2(t)$ y $I_3(t)$ con $\mathbb{E}(I_1(t)) = \mathbb{E}(I_2(t)) = \mathbb{E}(I_3(t)) = 0.1$ y $\mathrm{Var}(I_1(t)) = \mathrm{Var}(I_2(t)) = \mathrm{Var}(I_3(t)) = 0.1$.

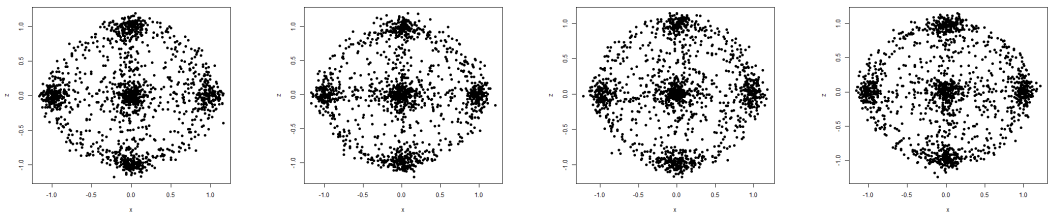


(a) $t = 2$ (b) $t = 10$ (c) $t = 15$ (d) $t = 20$

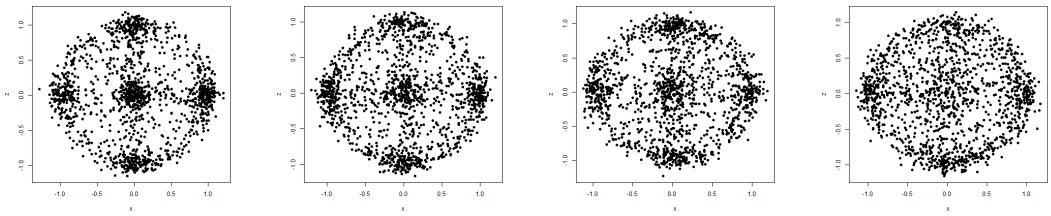


(e) $t = 25$ (f) $t = 30$ (g) $t = 40$ (h) $t = 50$

Figura C.17: 1500 datos sobre $\mathbb{S}^2$ con coeficientes IG $I_1(t)$,$I_2(t)$ y I_3 con $\mathbb{E}(I_1(t)) = 0.1 = \mathbb{E}(I_2(t)) = 0.1$ $\mathbb{E}(I_3(t)) = 1$ y $\text{Var}(I_1(t)) = \text{Var}(I_2(t)) = \text{Var}(I_3(t)) = 0.1$.

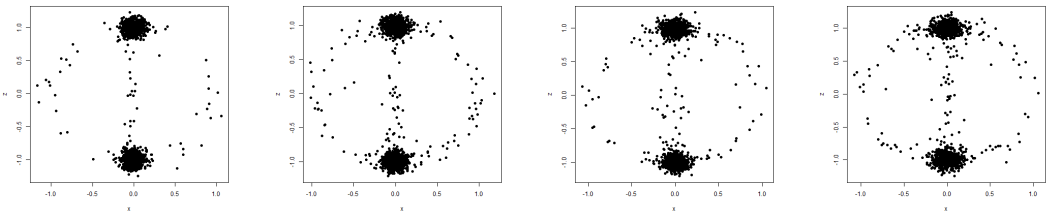


(a) $t = 2$ (b) $t = 10$ (c) $t = 25$ (d) $t = 50$

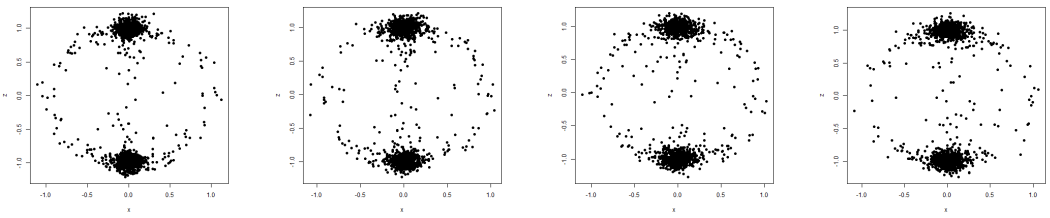


(e) $t = 100$ (f) $t = 300$ (g) $t = 500$ (h) $t = 1000$

Figura C.18: 1500 datos sobre $\mathbb{S}^2$ con coeficientes IG $I_1(t)$,$I_2(t)$ y $I_3(t)$ con $\mathbb{E}(I_1(t)) = \mathbb{E}(I_2(t)) = \mathbb{E}(I_3(t)) = 0.1$ y $\text{Var}(I_1(t)) = \text{Var}(I_2(t)) = \text{Var}(I_3(t)) = 10$.

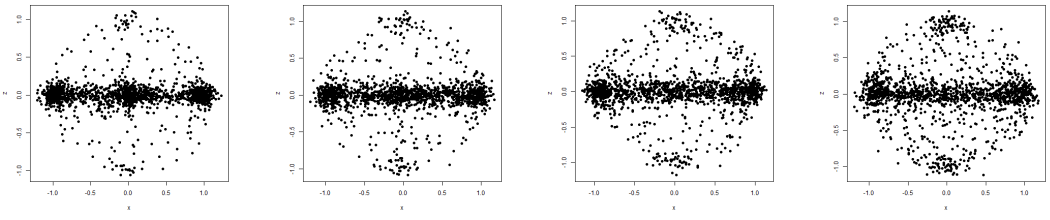


(a) $t = 2$ (b) $t = 10$ (c) $t = 25$ (d) $t = 50$

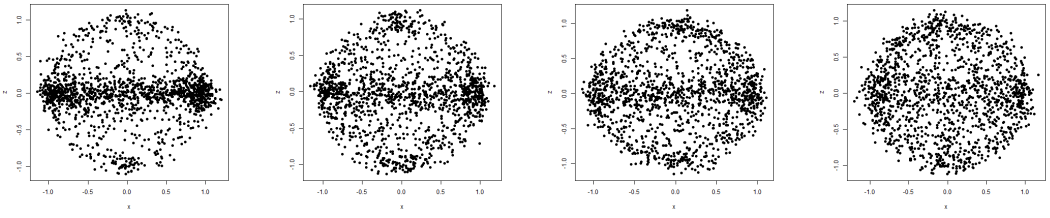


(e) $t = 100$ (f) $t = 300$ (g) $t = 500$ (h) $t = 1000$

Figura C.19: 1500 datos sobre $\mathbb{S}^2$ con coeficientes IG $I_1(t)$,$I_2(t)$ y $I_3(t)$ con $\mathbb{E}(I_1(t)) = \mathbb{E}(I_2(t)) = 0.1, \mathbb{E}(I_3(t)) = 2$ y $\mathrm{Var}(I_1(t)) = \mathrm{Var}(I_2(t)) = \mathrm{Var}(I_3(t)) = 10$.

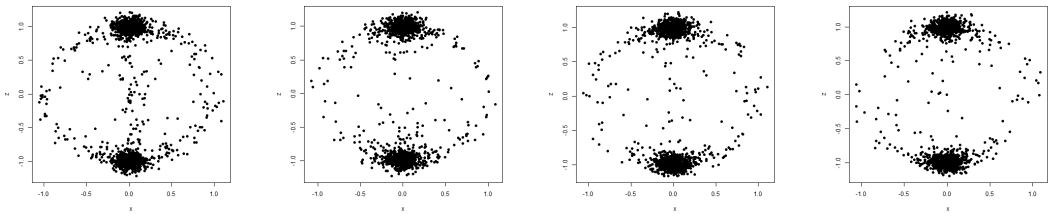


(a) $t = 2$ (b) $t = 10$ (c) $t = 25$ (d) $t = 50$

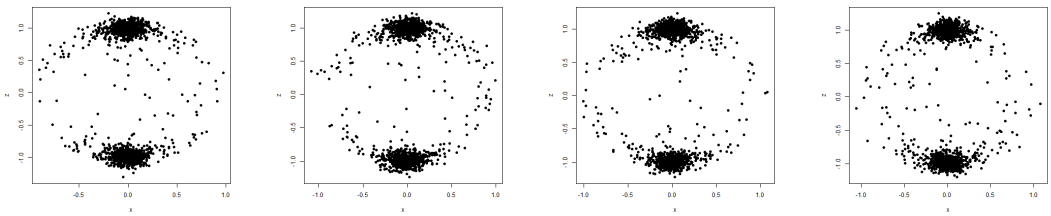


(e) $t = 100$ (f) $t = 300$ (g) $t = 500$ (h) $t = 1000$

Figura C.20: 1500 datos sobre $\mathbb{S}^2$ con coeficientes IG $I_1(t)$,$I_2(t)$ y $I_3(t)$ con $\mathbb{E}(I_1(t)) = \mathbb{E}(I_2(t)) = \mathbb{E}(I_3(t)) = 0.1$ y $\mathrm{Var}(I_1(t)) = \mathrm{Var}(I_2(t)) = 1, \mathrm{Var}(I_3(t)) = 10$.

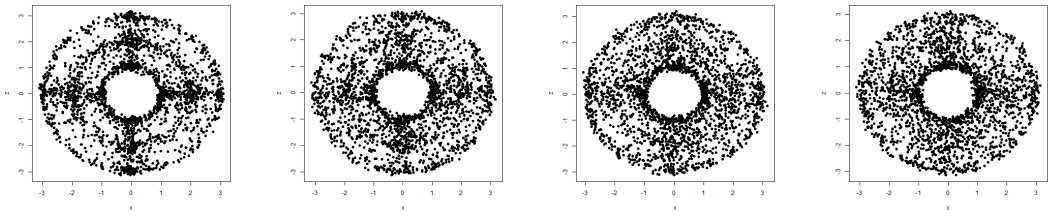


(a) $t = 2$ (b) $t = 10$ (c) $t = 25$ (d) $t = 50$

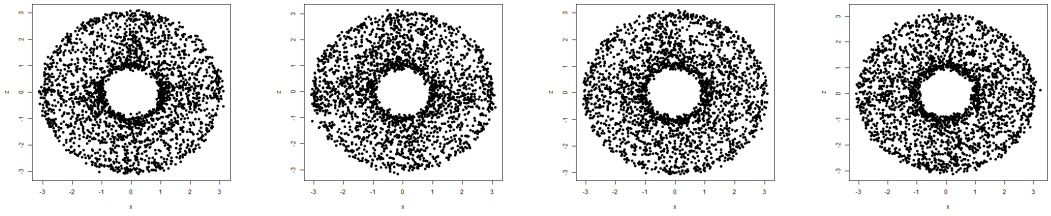


(e) $t = 100$ (f) $t = 300$ (g) $t = 500$ (h) $t = 1000$

Figura C.21: 1500 datos sobre $\mathbb{S}^2$ con coeficientes IG $I_1(t)$,$I_2(t)$ y $I_3(t)$ con $\mathbb{E}(I_1(t)) = \mathbb{E}(I_2(t)) = 0.1, \mathbb{E}(I_3(t)) = 2$ y $\text{Var}(I_1(t)) = \text{Var}(I_2(t)) = 0.1, = \text{Var}(I_3(t)) = 10$.

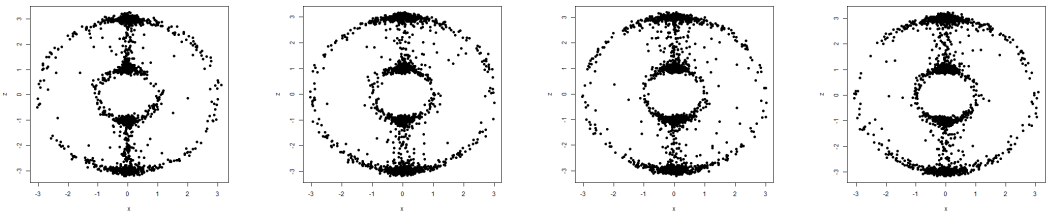


(a) $t = 2$ (b) $t = 10$ (c) $t = 15$ (d) $t = 20$

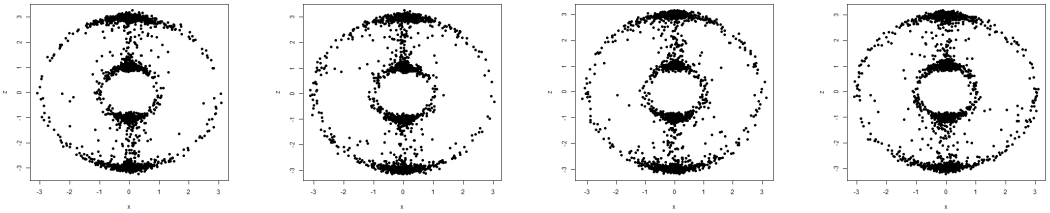


(e) $t = 25$ (f) $t = 30$ (g) $t = 40$ (h) $t = 50$

Figura C.22: 2500 datos sobre $\mathbb{T}^2$ con coeficientes IG $I_1(t)$,$I_2(t)$ y $I_3(t)$ con $\mathbb{E}(I_1(t)) = \mathbb{E}(I_2(t)) = \mathbb{E}(I_3(t)) = 0.1$ y $\text{Var}(I_1(t)) = \text{Var}(I_2(t)) = \text{Var}(I_3(t)) = 0.1$.

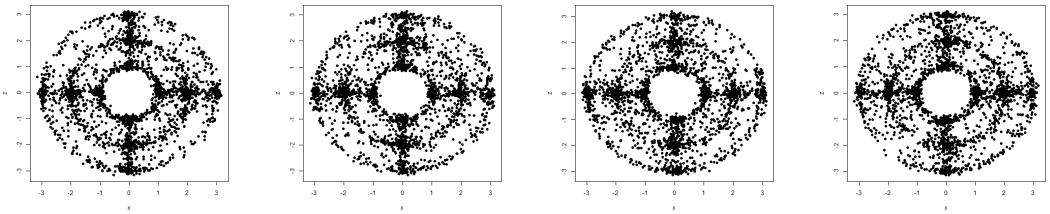


(a) $t = 2$ (b) $t = 10$ (c) $t = 15$ (d) $t = 20$

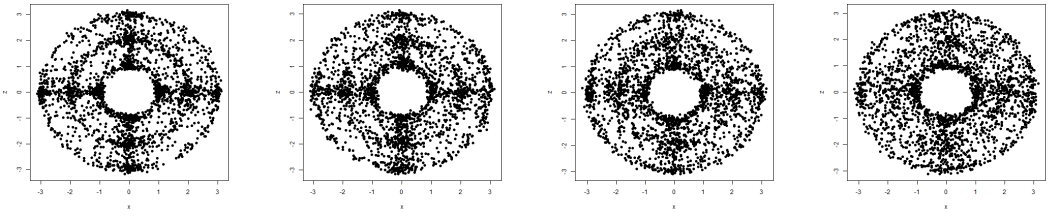


(e) $t = 25$ (f) $t = 30$ (g) $t = 40$ (h) $t = 50$

Figura C.23: 2500 datos sobre $\mathbb{T}^2$ con coeficientes IG $I_1(t)$,$I_2(t)$ y I_3 con $\mathbb{E}(I_1(t)) = 0.1 = \mathbb{E}(I_2(t)) = 0.1$ $\mathbb{E}(I_3(t)) = 1$ y $\mathrm{Var}(I_1(t)) = \mathrm{Var}(I_2(t)) = \mathrm{Var}(I_3(t)) = 0.1$.

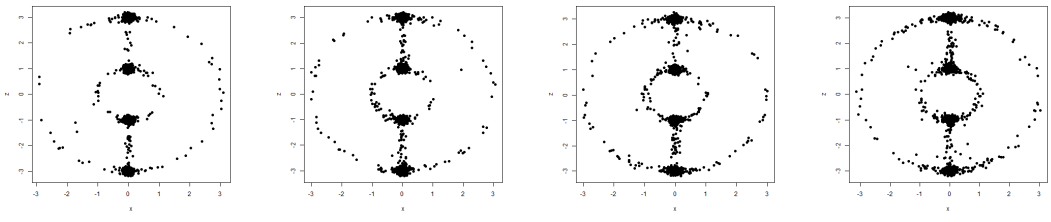


(a) $t = 2$ (b) $t = 10$ (c) $t = 25$ (d) $t = 50$

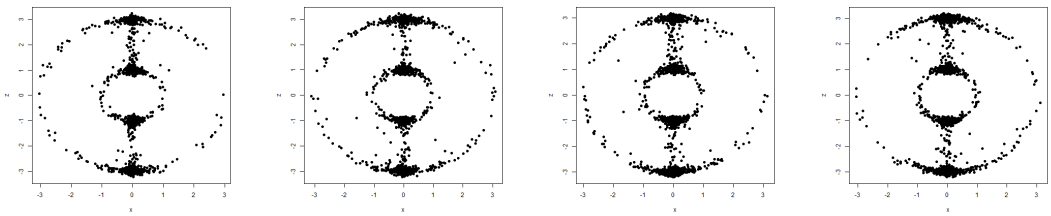


(e) $t = 100$ (f) $t = 300$ (g) $t = 500$ (h) $t = 1000$

Figura C.24: 2500 datos sobre $\mathbb{T}^2$ con coeficientes IG $I_1(t)$,$I_2(t)$ y $I_3(t)$ con $\mathbb{E}(I_1(t)) = \mathbb{E}(I_2(t)) = \mathbb{E}(I_3(t)) = 0.1$ y $\mathrm{Var}(I_1(t)) = \mathrm{Var}(I_2(t)) = \mathrm{Var}(I_3(t)) = 10$.

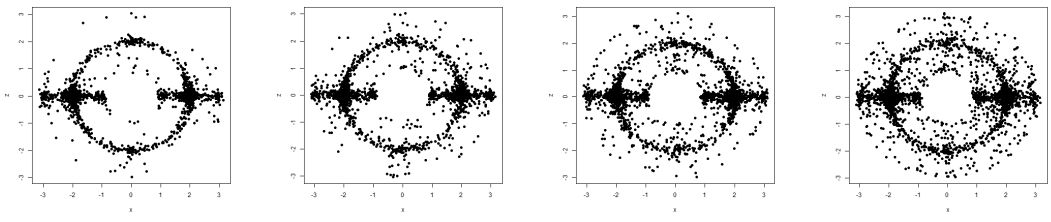


(a) $t = 2$ (b) $t = 10$ (c) $t = 25$ (d) $t = 50$

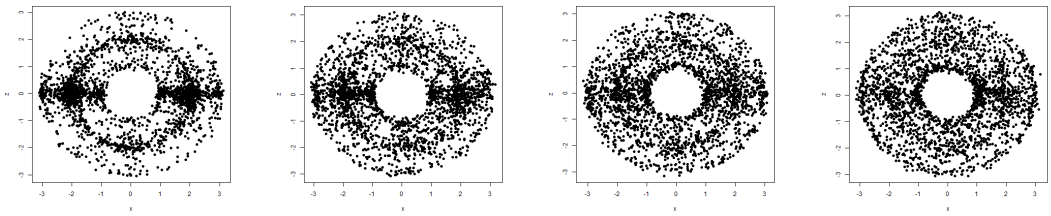


(e) $t = 100$ (f) $t = 300$ (g) $t = 500$ (h) $t = 1000$

Figura C.25: 2500 datos sobre $\mathbb{T}^2$ con coeficientes IG $I_1(t)$,$I_2(t)$ y $I_3(t)$ con $\mathbb{E}(I_1(t)) = \mathbb{E}(I_2(t)) = 0.1, \mathbb{E}(I_3(t)) = 2$ y $\text{Var}(I_1(t)) = \text{Var}(I_2(t)) = \text{Var}(I_3(t)) = 10$.

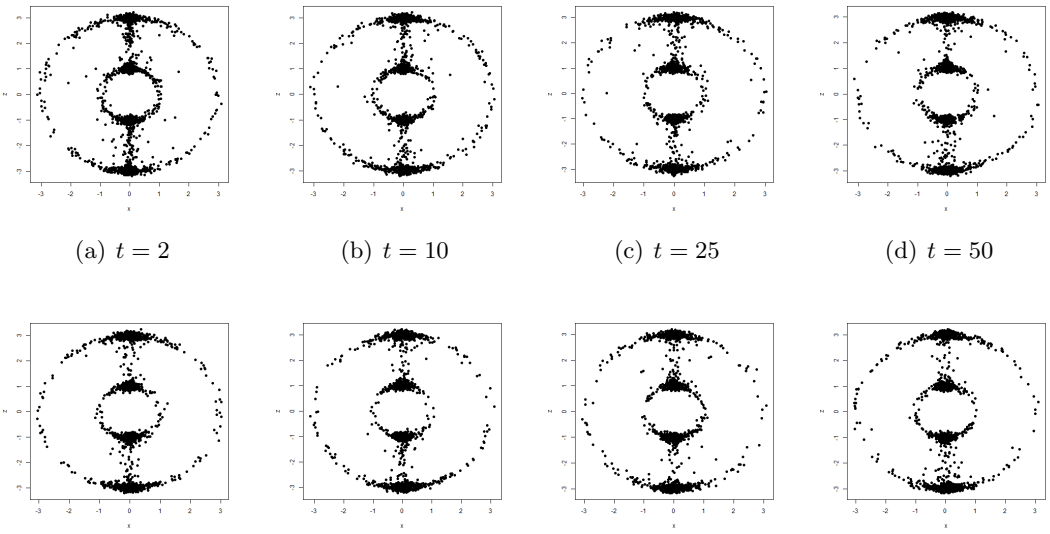


(a) $t = 2$ (b) $t = 10$ (c) $t = 25$ (d) $t = 50$



(e) $t = 100$ (f) $t = 300$ (g) $t = 500$ (h) $t = 1000$

Figura C.26: 2500 datos sobre $\mathbb{T}^2$ con coeficientes IG $I_1(t)$,$I_2(t)$ y $I_3(t)$ con $\mathbb{E}(I_1(t)) = \mathbb{E}(I_2(t)) = \mathbb{E}(I_3(t)) = 0.1$ y $\text{Var}(I_1(t)) = \text{Var}(I_2(t)) = 1, \text{Var}(I_3(t)) = 10$.

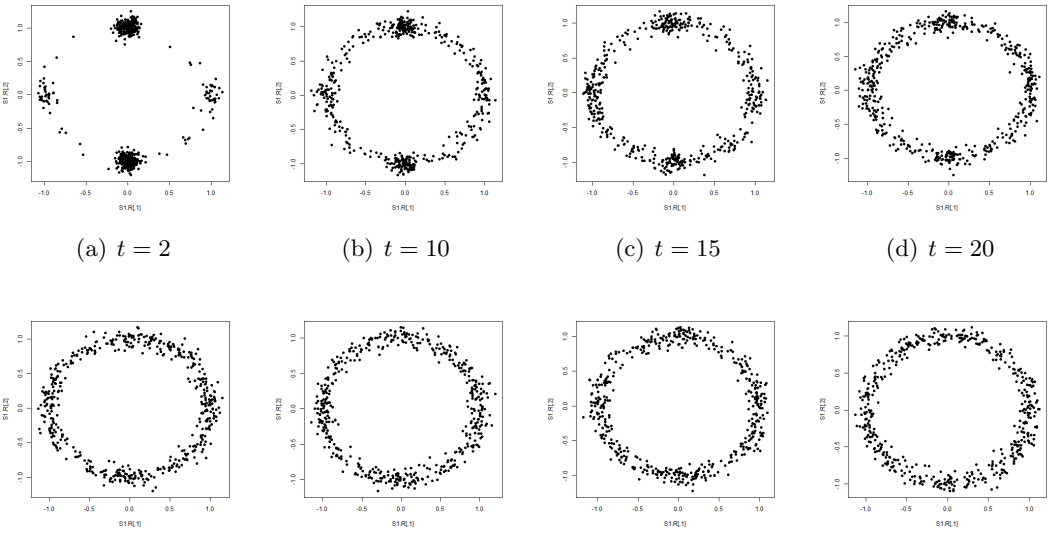


(a) $t = 2$ (b) $t = 10$ (c) $t = 25$ (d) $t = 50$

(e) $t = 100$ (f) $t = 300$ (g) $t = 500$ (h) $t = 1000$

Figura C.27: 2500 datos sobre $\mathbb{T}^2$ con coeficientes IG $I_1(t)$,$I_2(t)$ y $I_3(t)$ con $\mathbb{E}(I_1(t)) = \mathbb{E}(I_2(t)) = 0.1, \mathbb{E}(I_3(t)) = 2$ y $\mathrm{Var}(I_1(t)) = \mathrm{Var}(I_2(t)) = 0.1, = \mathrm{Var}(I_3(t)) = 10$.

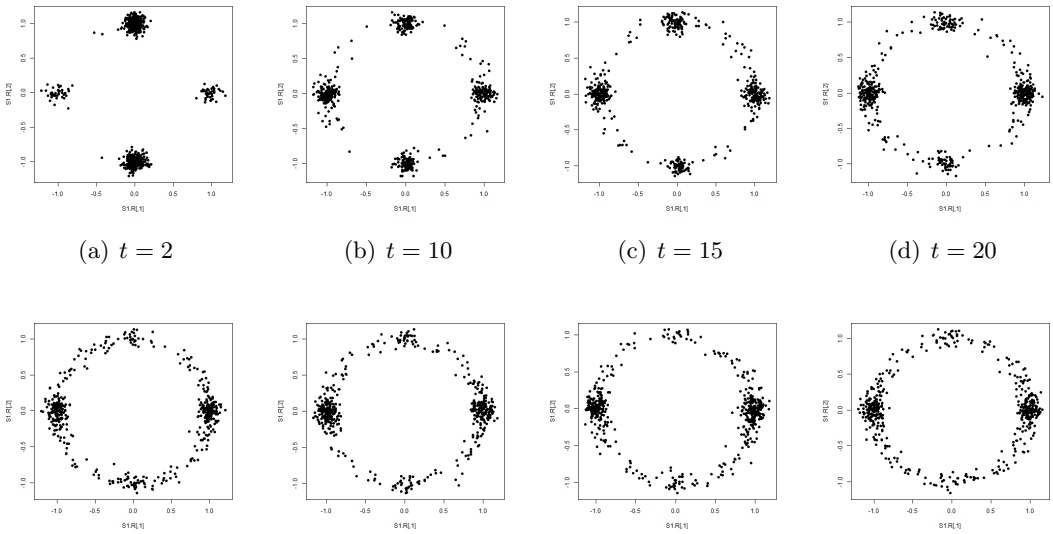
C.3. Perturbaciones mediante procesos Poisson/Inverso Gaussiano



(a) $t = 2$ (b) $t = 10$ (c) $t = 15$ (d) $t = 20$

(e) $t = 25$ (f) $t = 30$ (g) $t = 40$ (h) $t = 50$

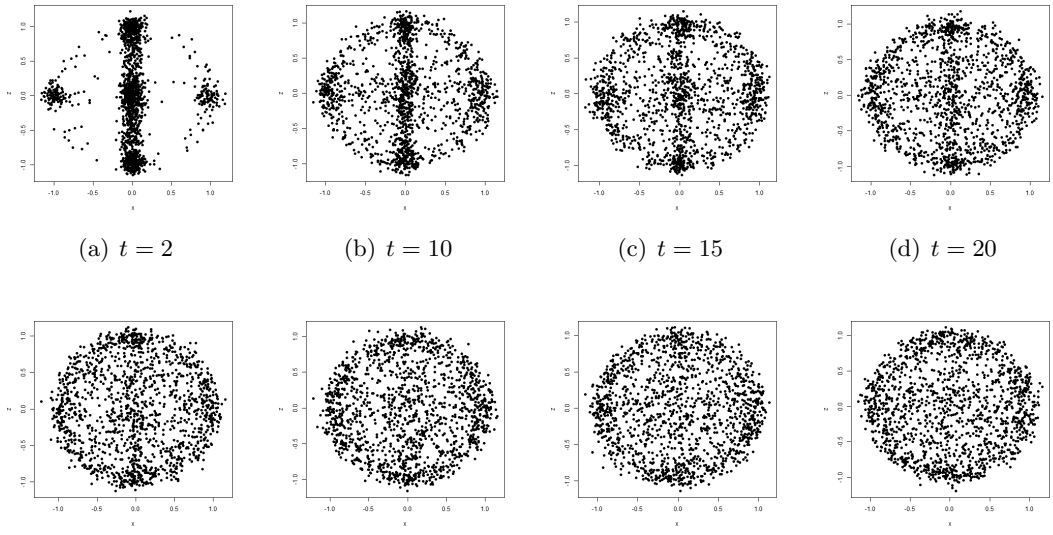
Figura C.28: 500 datos sobre $\mathbb{S}^1$ con coeficientes PP $N_1(t)$ e IG $I_2(t)$ con $\mathbb{E}(N_1(t)) = \mathbb{E}(I_2(t)) = 0.1$ y $\mathrm{Var}(N_1(t)) = \mathrm{Var}(I_2(t)) = 0.1$.



(a) $t = 2$ (b) $t = 10$ (c) $t = 15$ (d) $t = 20$

(e) $t = 25$ (f) $t = 30$ (g) $t = 40$ (h) $t = 50$

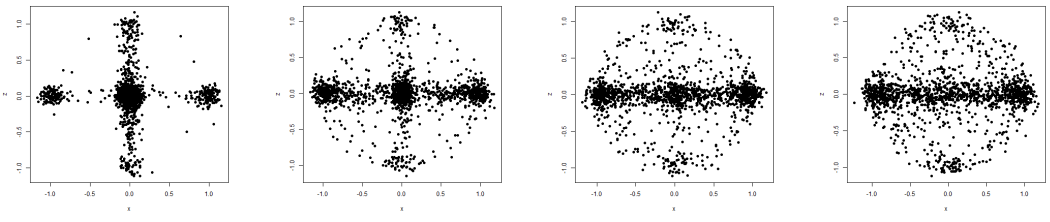
Figura C.29: 500 datos sobre $\mathbb{S}^1$ con coeficientes PP $N_1(t)$ e IG $I_2(t)$ con $\mathbb{E}(N_1(t)) = \mathbb{E}(I_2(t)) = 0.1$ y $\text{Var}(N_1(t))0.1, \text{Var}(I_2(t)) = 10$.



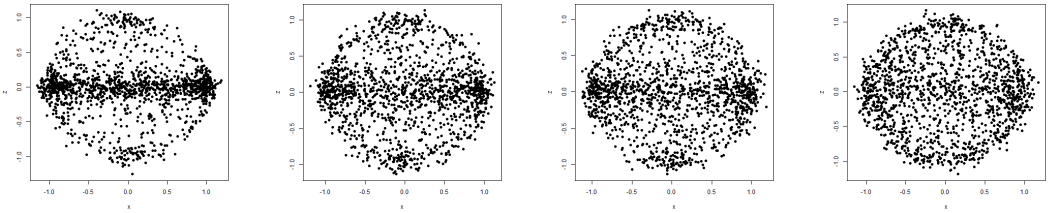
(a) $t = 2$ (b) $t = 10$ (c) $t = 15$ (d) $t = 20$

(e) $t = 25$ (f) $t = 30$ (g) $t = 40$ (h) $t = 50$

Figura C.30: 500 datos sobre $\mathbb{S}^2$ con coeficientes PP $N_1(t)$ e IG $I_2(t), I_3(t)$ con $\mathbb{E}(N_1(t)) = \mathbb{E}(I_2(t)) = \mathbb{E}(I_3(t)) = 0.1$ y $\text{Var}(N_1(t)) = \text{Var}(I_2(t)) = \text{Var}(I_3(t)) = 0.1$.

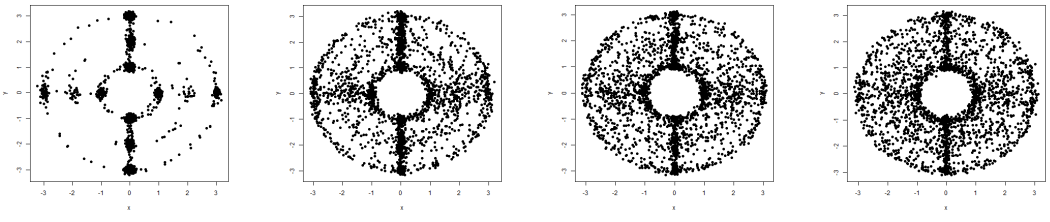


(a) $t = 2$ (b) $t = 10$ (c) $t = 25$ (d) $t = 50$

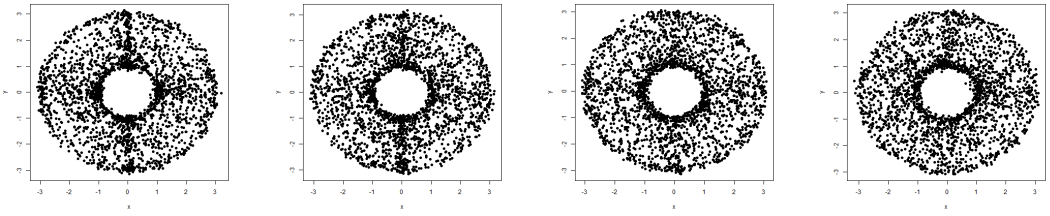


(e) $t = 100$ (f) $t = 300$ (g) $t = 500$ (h) $t = 1000$

Figura C.31: 1500 datos sobre $\mathbb{S}^2$ con coeficientes PP $N_1(t)$ e IG $I_2(t)$, $I_3(t)$ con $\mathbb{E}(N_1(t)) = \mathbb{E}(I_2(t)) = \mathbb{E}(I_3(t)) = 0.1$ y $\mathrm{Var}(N_1(t)) = \mathrm{Var}(I_2(t))0.1, \mathrm{Var}(I_3(t)) = 10$.

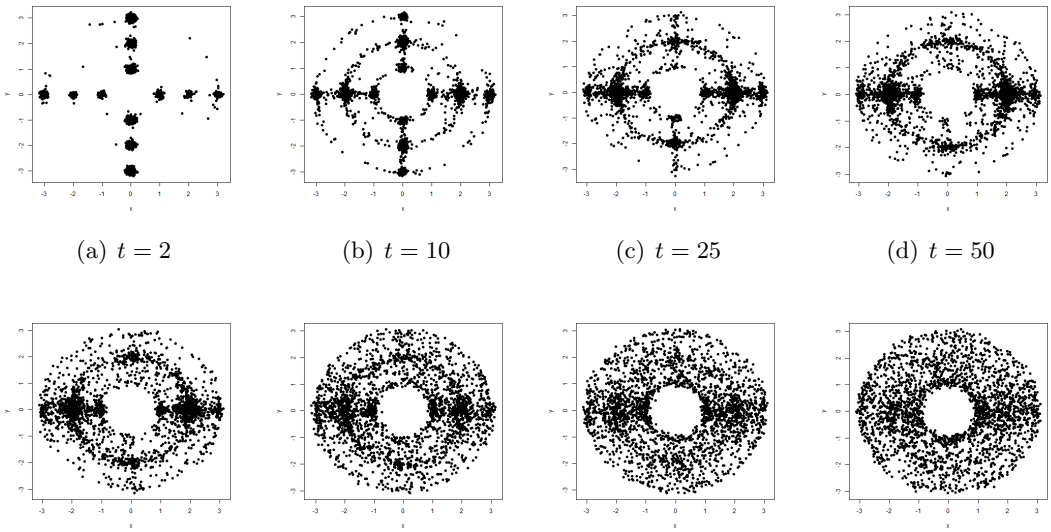


(a) $t = 2$ (b) $t = 10$ (c) $t = 15$ (d) $t = 20$



(e) $t = 25$ (f) $t = 30$ (g) $t = 40$ (h) $t = 50$

Figura C.32: 2500 datos sobre $\mathbb{T}^2$ con coeficientes PP $N_1(t)$ e IG $I_2(t), I_3(t)$ con $\mathbb{E}(N_1(t)) = \mathbb{E}(I_2(t)) = \mathbb{E}(I_3(t)) = 0.1$ y $\mathrm{Var}(N_1(t)) = \mathrm{Var}(I_2(t)) = \mathrm{Var}(I_3(t)) = 0.1$.



(a) $t = 2$ (b) $t = 10$ (c) $t = 25$ (d) $t = 50$

(e) $t = 100$ (f) $t = 300$ (g) $t = 500$ (h) $t = 1000$

Figura C.33: 2500 datos sobre $\mathbb{T}^2$ con coeficientes PP $N_1(t)$ e IG $I_2(t)$, $I_3(t)$ con $\mathbb{E}(N_1(t)) = \mathbb{E}(I_2(t)) = \mathbb{E}(I_3(t)) = 0.1$ y $\text{Var}(N_1(t)) = \text{Var}(I_2(t))0.1, \text{Var}(I_3(t)) = 10$.

Bibliografía

Arsuaga, J., Baas, N. A., DeWoskin, D., Mizuno, H., Pankov, A., y Park, C. (2012). Topological analysis of gene expression arrays identifies high risk molecular subtypes in breast cancer. *Applicable Algebra in Engineering, Communication and Computing*, *23*(1-2), 3–15. (Citado en página 1.)

Bak, A. (2016). Web del autor. https://www.ayasdi.com/blog/author/anthony-bak/. Accesado: 03/11/2016. (Citado en página 77.)

Biscay, R., Nakamura, M., Pérez Abreu, V., y Reveles, F. (2016). *Persistencia, Probabilidad e Inferencia Estadística para Análisis Topológico de Datos*. CIMAT. (Citado páginas 3, 8, 30 and 45.)

Borsuk, K. (1948). On the imbedding of systems of compacta in simplicial complexes. *Fundamenta Mathematicae*, *35*(1), 217–234. (Citado en página 34.)

Bowman, G. R., Huang, X., Yao, Y., Sun, J., Carlsson, G., Guibas, L. J., y Pande, V. S. (2008). Structural insight into rna hairpin folding intermediates. *Journal of the American Chemical Society*, *130*(30), 9676–9678. (Citado en página 1.)

Brown, J. y Gedeon, T. (2012). Structure of the afferent terminals in terminal ganglion of a cricket and persistent homology. *PloS one*, *7*(5), e37278. (Citado en página 1.)

Bubenik, P. (2015). Statistical topological data analysis using persistence landscapes. *Journal of Machine Learning Research*, *16*(1), 77–102. (Citado en página 40.)

Bubenik, P., Carlsson, G., Kim, P. T., y Luo, Z.-M. (2010). Statistical topology via morse theory persistence and nonparametric estimation. *Algebraic methods in statistics and probability II*, *516*, 75–92. (Citado páginas 71 and 119.)

Campbell, J. (1980). On Temme's algorithm for the modified Bessel function of the third kind. *ACM Transactions on Mathematical Software (TOMS)*, *6*(4), 581–586. (Citado en página 17.)

Carlsson, G., Ishkhanov, T., De Silva, V., y Zomorodian, A. (2008). On the local behavior of spaces of natural images. *International Journal of Computer Vision*, *76*(1), 1–12. (Citado en página 1.)

Chan, J. M., Carlsson, G., y Rabadan, R. (2013). Topology of viral evolution. *Proceedings of the National Academy of Sciences*, *110*(46), 18566–18571. (Citado en página 1.)

Dabaghian, Y., Mémoli, F., Frank, L., y Carlsson, G. (2012). A topological paradigm for hippocampal spatial map formation using persistent homology. *PLoS Comput Biol*, *8*(8), e1002581. (Citado en página 1.)

De Silva, V. y Carlsson, G. (2004). Topological estimation using witness complexes. In *Proc. Sympos. Point-Based Graphics*, (pp. 157–166). (Citado páginas 2, 6, 65, 66, 68 and 118.)

De Silva, V. y Ghrist, R. (2007). Coverage in sensor networks via persistent homology. *Algebraic & Geometric Topology*, *7*(1), 339–358. (Citado en página 1.)

DeWoskin, D., Climent, J., Cruz-White, I., Vazquez, M., Park, C., y Arsuaga, J. (2010). Applications of computational homology to the analysis of treatment response in breast cancer patients. *Topology and its Applications*, *157*(1), 157–164. (Citado en página 1.)

Dey, T. K., Shi, D., y Wang, Y. (2016). Simba: An efficient tool for approximating rips-filtration persistence via simplicial batch-collapse. In *arXiv preprint arXiv:1609.07517*. (Citado en página 119.)

Edelsbrunner, H. y Harer, J. (2010). *Computational Topology: An Introduction*. American Mathematical Society. (Citado páginas 30, 41 and 42.)

Edelsbrunner, H., Kirkpatrick, D., y Seidel, R. (1983). On the shape of a set of points in the plane. *IEEE Transactions on information theory*, *29*(4), 551–559. (Citado páginas 57, 58 and 59.)

Edelsbrunner, H. y Morozov, D. (2012). Persistent homology: theory and practice. In *Proceedings of the European Congress of Mathematics*, (pp. 31–50). (Citado en página 39.)

Edelsbrunner, H. y Mücke, E. P. (1994). Three-dimensional alpha shapes. *ACM Transactions on Graphics (TOG)*, *13*(1), 43–72. (Citado páginas 2 and 57.)

Efron, B. y Tibshirani, R. J. (1994). *An Introduction to the Bootstrap*. CRC press. (Citado en página 38.)

Emmett, K. J. y Rabadan, R. (2014). Characterizing scales of genetic recombination and antibiotic resistance in pathogenic bacteria using topological data analysis. In *International Conference on Brain Informatics and Health*, (pp. 540–551). Springer. (Citado en página 1.)

Fasy, B., Kim, J., Lecci, F., y Maria, C. (2014). Introduction to the r package tda. In *arXiv preprint arXiv:1411.1830*. (Citado páginas 45 and 92.)

Gamble, J. y Heo, G. (2010). Exploring uses of persistent homology for statistical analysis of landmark-based shape data. *Journal of Multivariate Analysis*, *101*(9), 2184–2199. (Citado en página 1.)

González Cucurachi, V. A. (2016). *Aspectos Estadísticos en Análisis Topológico de Datos y una Aplicación en Ecología*. CIMAT. (Citado páginas 1 and 38.)

Guibas, L. J. y Oudot, S. Y. (2008). Reconstruction using witness complexes. *Discrete & Computational Geometry*, *40*(3), 325–356. (Citado en página 2.)

Hennigan, R. (2015). A fast simplicial complex construction for computing the persistent homology of very large and high dimensional data sets. In *Manuscript*. (Citado páginas 3, 60, 86 and 87.)

Hernandez-Stumpfhauser, D., Breidt, F. J., y van der Woerd, M. J. (2016). The general projected normal distribution of arbitrary dimension: Modeling and bayesian inference. (Citado en página 4.)

Herstein, I. N. (1988). *Álgebra Moderna*. México, MX: Trillas. (Citado en página 25.)

Horak, D., Maletić, S., y Rajković, M. (2009). Persistent homology of complex networks. *Journal of Statistical Mechanics: Theory and Experiment*, *2009*(03), P03034. (Citado en página 1.)

Ibarra Rodríguez, J. M. (2016). *Modelos de Homología Persistente en Filogenética*. CIMAT. (Citado en página 1.)

Jung, H. W. E. (1899). *Über die kleinste kugel die eine räumliche figur einschliesst...* Marburg. (Citado en página 58.)

Kyprianou, A. (2006). *Introductory lectures on fluctuations of Lévy processes with applications*. Springer Science & Business Media. (Citado en página 14.)

Milnor, J. (1963). *Morse Theory*. University Press, Princeton. (Citado en página 2.)

Morse, M. (1934). *The Calculus of Variations in the Large*, volume 18. American Mathematical Soc. (Citado en página 2.)

Müllner, D. y Babu, A. (2013). Python mapper: An open-source toolchain for data exploration, analysis and visualization. http://danifold.net/mapper. Accesado: 29/08/2016. (Citado páginas 3 and 80.)

Nicolau, M., Levine, A. J., y Carlsson, G. (2011). Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival. *Proceedings of the National Academy of Sciences*, *108*(17), 7265–7270. (Citado en página 1.)

Otter, N., Porter, M. A., Tillmann, U., Grindrod, P., y Harrington, H. A. (2015). A roadmap for the computation of persistent homology. In *arXiv preprint arXiv:1506.08903*. (Citado páginas 5, 89, 90, 91, 92 and 93.)

Romano, D., Nicolau, M., Quintin, E.-M., Mazaika, P. K., Lightbody, A. A., Cody Hazlett, H., Piven, J., Carlsson, G., y Reiss, A. L. (2014). Topological methods reveal high and low functioning neuro-phenotypes within fragile x syndrome. *Human brain mapping*, *35*(9), 4904–4915. (Citado en página 1.)

Rote, G. (1991). Computing the minimum hausdorff distance between two point sets on a line under translation. *Information Processing Letters*, *38*(3), 123–127. (Citado en página 27.)

Seemann, L., Shulman, J., y Gunaratne, G. H. (2012). A robust topology-based algorithm for gene expression profiling. *ISRN Bioinformatics*, *2012*. (Citado en página 1.)

Singh, G., Mémoli, F., y Carlsson, G. E. (2007). Topological methods for the analysis of high dimensional data sets and 3d object recognition. In *SPBG*, (pp. 91–100). (Citado páginas 2, 76, 77 and 78.)

Singh, G., Memoli, F., Ishkhanov, T., Sapiro, G., Carlsson, G., y Ringach, D. L. (2008). Topological analysis of population activity in visual cortex. *Journal of Vision*, *8*(8), 11–11. (Citado en página 1.)

Tsybakov, A. B. (2009). *Introduction to Nonparametric Estimation*. Springer Series in Statistics. Springer, New York. (Citado en página 44.)

van Veen, H. (2015). Kepler mapper. https://github.com/MLWave/kepler-mapper. Accesado: 10/09/2016. (Citado páginas 3 and 82.)

Willard, S. (1970). *General Topology*. Courier Corporation. (Citado páginas 25, 26 and 27.)

Zomorodian, A. J. (2005). *Topology for Computing*, volume 16. Cambridge University Press. (Citado en página 116.)