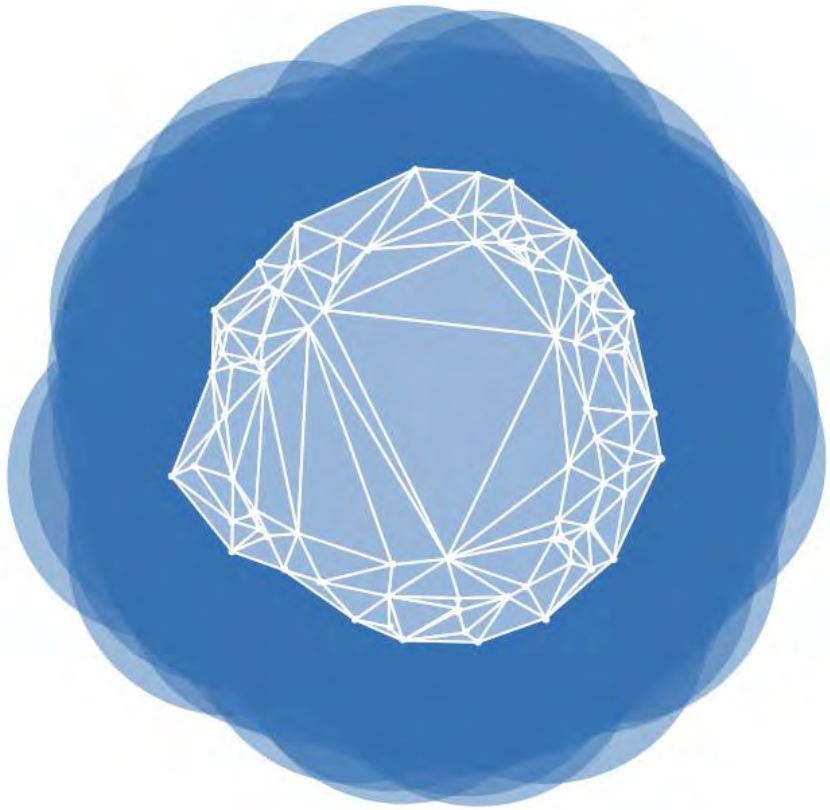
Aspectos computacionales del "Análisis Topológico de Datos"

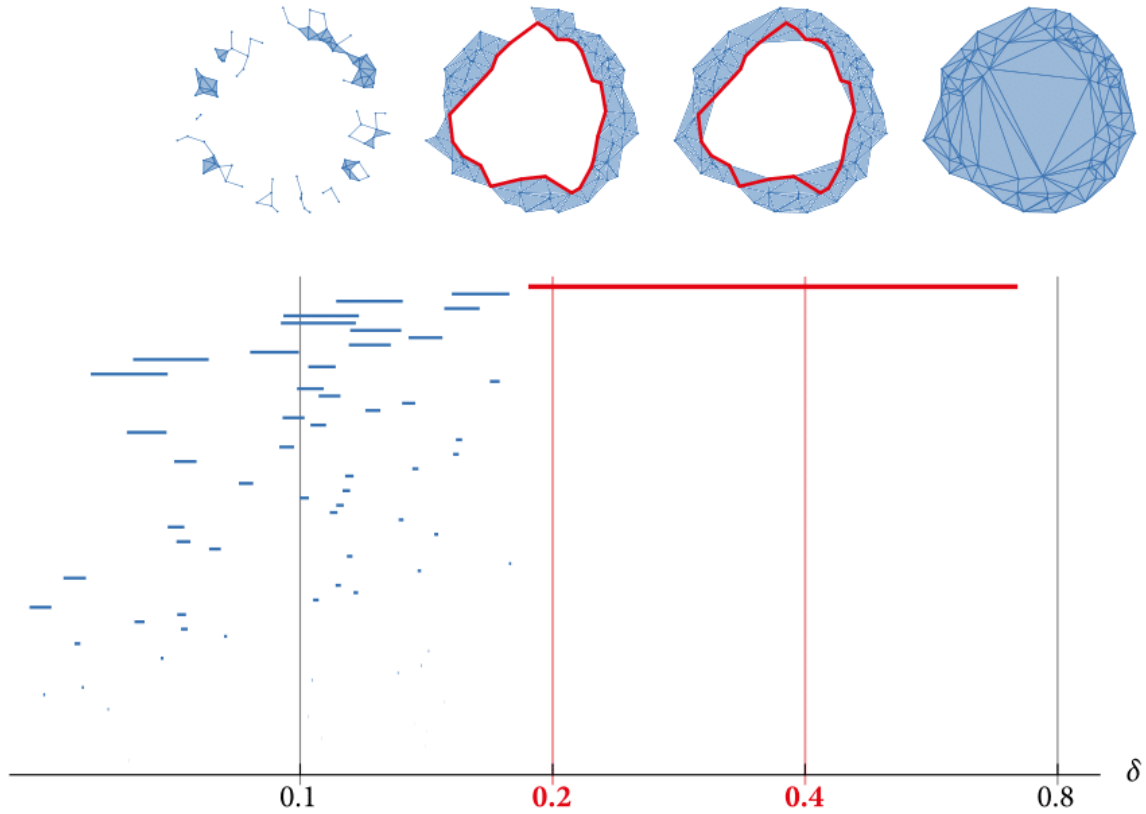
FRANCISCO VALENTE CASTRO

Homología Persistente



http://ulrich-bauer.org/ripser-talk.pdf

Homología Persistente



http://ulrich-bauer.org/ripser-talk.pdf

Homología Persistente (Complejo Simplicial)

- Sea $\boldsymbol{K} = \{\, \boldsymbol{\sigma_1}, \ldots, \boldsymbol{\sigma_n} \,\}$ un complejo simplicial de ***dimensión d*** y $\boldsymbol{G}$ ***un campo de coeficientes*** $(\, usaremos\ \mathbb{Z}_2\ por\ simplicidad\,)$**.**
- Consideremos un ordenamiento de los simplejos tal que $\boldsymbol{i} \leq \boldsymbol{n},\ \boldsymbol{K_i} \coloneqq \{\boldsymbol{\sigma_1}, \ldots, \boldsymbol{\sigma_i}\}$ es también un complejo simplicial.
- La cadena $\emptyset = \boldsymbol{K_0} \subset \boldsymbol{K_1} \subset \ldots \boldsymbol{K_n} = \boldsymbol{K}$ es llamada una ***filtración de K.***
- Normalmente, una filtración es definida siguiendo una **función** $\boldsymbol{f} : \boldsymbol{K} \rightarrow \mathbb{R}$ que ordena los simplejos de K siguiendo el valor de la función.

Homología Persistente (d-cadenas)

Una ***d-cadena*** es una función del conjunto de ***d-simplejos*** dotados de una orientación a **F**, $c : K \to F$ tal que :

a) $c(\sigma) = -c(\sigma')$ si σ y σ' *son orientaciones opuestas del mismo simplejo,*

b) $c(\sigma) = 0$ *para todos los simplejos menos un conjunto finito.*

Así que podemos sumar las cadenas sumando sus valores ; el conjunto de **d-cadenas con +** forma un grupo abeliano libre $\mathbf{C}_d(\mathrm{K}, \mathrm{G}) = \langle \boldsymbol{\sigma} \rangle_{\boldsymbol{\sigma} \in \boldsymbol{K^d}}$ generado por los **d-simplejos de K.**

Homología Persistente (Operador Frontera)

Ahora definamos el homomorfismo:

$$\partial_d : \mathbf{C}_d(\mathbf{K}, \mathrm{G}) \to \mathbf{C}_{d-1}(\mathbf{K}, \mathrm{G})$$

llamado el **"operador frontera"**, siendo $\partial_d(\ \sigma\ )$ la suma **"orientada"** de las caras de dimensión **(d – 1)** de nuestro simplejo $\boldsymbol{\sigma}$.

Lema. (***Lema Fundamental de la Homología*** **).**

$$\partial_d \circ \partial_{d+1} = 0$$

Al kernel de $\partial_d : \mathbf{C}_d(\mathbf{K}, \mathrm{G}) \to \mathbf{C}_{d-1}(\mathbf{K}, \mathrm{G})$ lo llamamos el grupo de **d-ciclos** y es denotado por $\mathbf{Z}_d(\mathbf{K}, \mathrm{G})$.

A la imagen de $\partial_{d+1} : \mathbf{C}_{d+1}(\mathbf{K}, \mathrm{G}) \to \mathbf{C}_d(\mathbf{K}, \mathrm{G})$ lo llamamos el grupo de **d-fronteras** y lo denotamos por $\mathbf{B}_d(\mathbf{K}, \mathrm{G})$.

Por el lema, es claro que $\mathbf{B}_d(\mathbf{K}, \mathrm{G}) \subseteq \mathbf{Z}_d(\mathbf{K}, \mathrm{G})$.

Homología Persistente (Grupo de Homología)

Ahora definamos :

$$\mathbf{H}_d(\mathbf{K}, \mathrm{G}) = \mathbf{Z}_d(\mathbf{K}, \mathrm{G})/\mathbf{B}_d(\mathbf{K}, \mathrm{G})$$

como **d-esimo grupo de homología** de K *con coeficientes en* G**.**

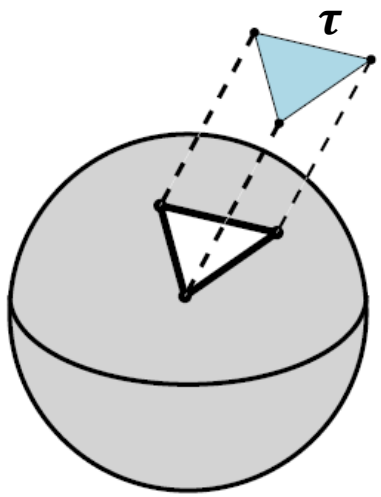
Como los coeficientes de los grupos mencionados están en **G**, los grupos $\mathbf{C}_d(\mathbf{K}, \mathrm{G})$, $\mathbf{B}_d(\mathbf{K}, \mathrm{G})$, $\mathbf{Z}_d(\mathbf{K}, \mathrm{G})$, $\mathbf{H}_d(\mathbf{K}, \mathrm{G})$ también son espacios vectoriales sobre G ($\mathbb{Z}_\mathbf{2}$)

Homología Persistente

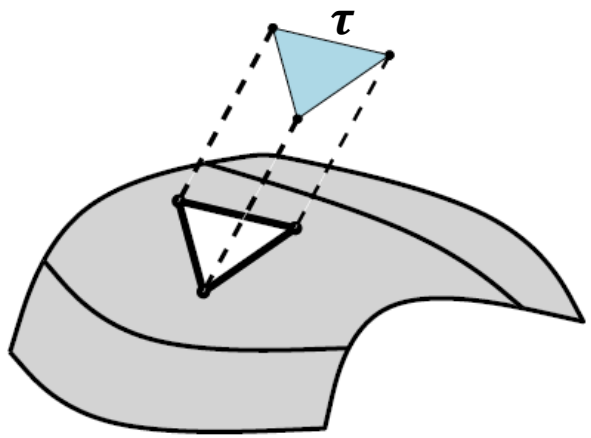
Estudiemos un poco que ocurre con una inclusión elemental $\boldsymbol{K} \xrightarrow{\tau} \boldsymbol{L}$ en los grupos de homología. Suponiendo que $\boldsymbol{\tau}$ **es un d-simplejo**, tenemos el siguiente resultado

---

Lema. *Si existe un ciclo* $c \in \boldsymbol{C_d}(\boldsymbol{L})$ *con* $c(\boldsymbol{\tau}) \neq 0$, *entonces* $dim\, \boldsymbol{H_d}(\boldsymbol{L}) = dim\, \boldsymbol{H_d}(\boldsymbol{K}) + 1$ *y los otros grupos de homología no cambian. Decimos que* $\boldsymbol{\tau}$ *es un* ***creador ó un simplejo positivo****. De lo contrario,* $dim\, \boldsymbol{H_{d-1}}(\boldsymbol{L}) = dim\, \boldsymbol{H_{d-1}}(\boldsymbol{K}) - 1$ *y los otros grupos de homología no cambian. Decimos que* $\boldsymbol{\tau}$ *es un* ***destructor ó un simplejo negativo.***



Creación de un 2-ciclo



Destrucción de un 1-ciclo

Algoritmo Homología Persistente (Incremental)

Marco :

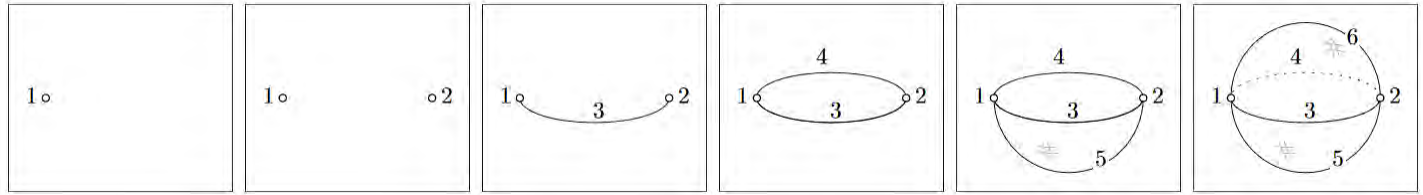
- N puntos en $\mathbb{R}_3$
- Homología persistente $H_d(Rips_t(X),\ \mathbb{Z}_2)\ en\ las\ dimensiones\ d \leq k$
- La filtración obtenida es $\boldsymbol{K_i} \coloneqq \{\boldsymbol{\sigma_1}, \ldots, \boldsymbol{\sigma_i}\}$, $\mathbf{i = 1}, \ldots, \mathbf{m}$ **(** siendo **m** el número de simplejos **)**

Notación :

- $\boldsymbol{D}$: Matriz frontera de la filtración $Rips_t(X)$
- $\boldsymbol{R_i}$: i-esima columna de R

Algoritmo :

- $\boldsymbol{\beta_0 = \beta_1 = \beta_2 = 0}$
- **Para cada** $j = 1, \ldots, m$
  - $\boldsymbol{k = dim\ \sigma_i\ -\ 1\ ;}$
  - $\boldsymbol{Si\ \sigma_i}\ pertence\ a$ un (k+1)-ciclo en $\boldsymbol{K^k}$
    - $Entonces\ \boldsymbol{\beta_{k+1} = \beta_{k+1} + 1}$
    - $Sino\ \boldsymbol{\beta_k = \beta_k - 1}$
- $\boldsymbol{End}\ (\boldsymbol{\beta_0, \beta_1, \beta_2})$



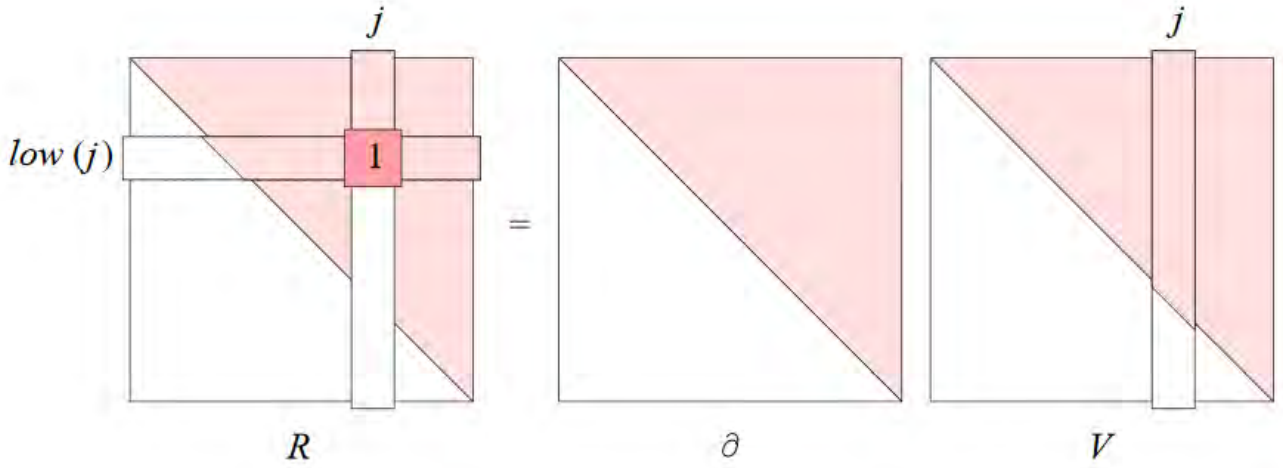
Algoritmo (Reducción Matricial)

Marco :

- Espacio métrico finito X, n puntos
- Homología persistente $H_d(\, Rips_t(\, X\,),\ \mathbb{Z}_2\,)\ en\ las\ dimensiones\ d \leq k$
- La filtración obtenida es $\boldsymbol{K_i} \coloneqq \{\boldsymbol{\sigma_1}, \ldots, \boldsymbol{\sigma_i}\}$, $\mathbf{i = 1, \ldots, m}$ **(** siendo **m** el número de simplejos **)**

Notación :

- $\boldsymbol{D}$: Matriz frontera de la filtración $Rips_t(\, X\,)$
- $\boldsymbol{R_i}$: i-esima columna de R



Algoritmo :

- $\boldsymbol{R} = D, \boldsymbol{V} = I, \boldsymbol{L} = [0, \ldots, 0]$
- $\boldsymbol{for}\ j = 1, \ldots, m\ \boldsymbol{do}$
  - $\boldsymbol{while}\ R_j \neq 0 \quad \boldsymbol{y} \quad \boldsymbol{L}\big[\, low\big(\, R_j\,\big)\big] \neq 0\ \boldsymbol{do}$
    - $R_j \leftarrow R_j + R_{L[low(R_j)]}$
    - $V_j \leftarrow V_j + V_{L[low(R_j)]}$
  - $\boldsymbol{if}\ R_j \neq 0\ \boldsymbol{then}\ \boldsymbol{L}\big[\, low\big(\, R_j\,\big)\big] \leftarrow j$
- $\boldsymbol{return}\ \boldsymbol{R}$

Algoritmo : ¿ Qué información obtenemos ?

Para la matriz frontera reducida $R = D * V$, llamemos

- $P = \{ i : R_i = 0 \}$ indices positivos (simplejo creador)
- $N = \{ j : R_j \neq 0 \}$ indices negativos (simplejo destructor)
- $E = P \setminus \boldsymbol{pivots}\, R$ indices escenciales (creador nunca destrido)

Entonces

Obtenemos la siguiente información ,

- Intervalos de persistencia : $\{ [i , j) \mid i = \boldsymbol{pivot}\, R_j \} \cup \{ [i, \infty) \mid i \in E \}$

¿ Cómo se ha optimizado ?

गुढी GUDHI Geometry Understanding in Higher Dimensions

Creador : **Clément Maria,** Jean-Daniel Boissonnat, Marc Glisse, Mariette Yvinec

- Simplex Tree
- Compressed Annotation Matrix
- Calcular Cohomología

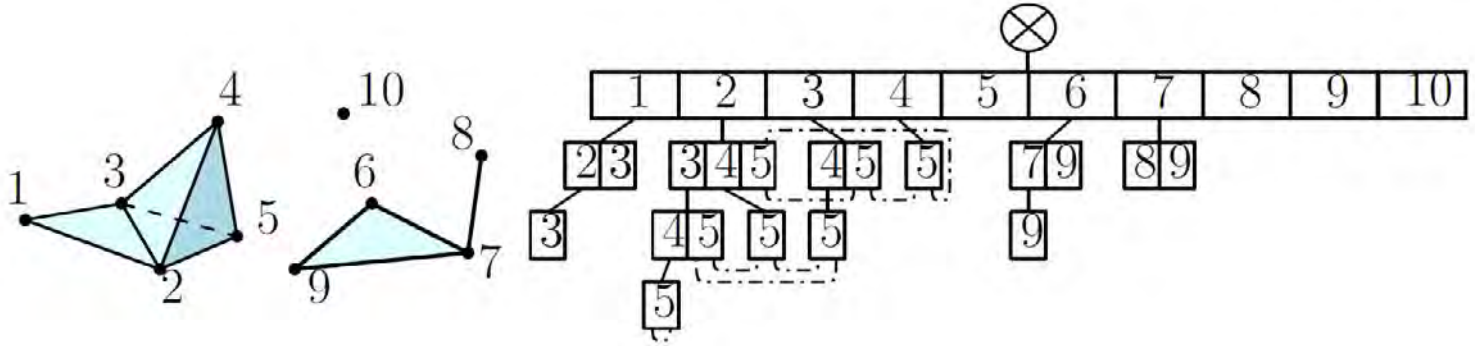
- En algunos casos, calcular la **“Cohomología Persistente”** de una filtración es mucho más rápido que la **“Homología Persistente”** (de Silva et al., 2011), como el caso partíclar de las filtraciones de Vietor-Rips.

Ripser (http://ripser.org)

- Creador : **Ulrich Bauer**

Las optimizaciones echas basadas en estas ideas:
- Eliminar columnas innecesarias [Chen, Kerber]
- Calcular Cohomología [de Silva et al.]
- Reducción de matriz implicitamente
- Pares Apparentes y Emergentes en la filtración

GUDHI (Simplex Tree)



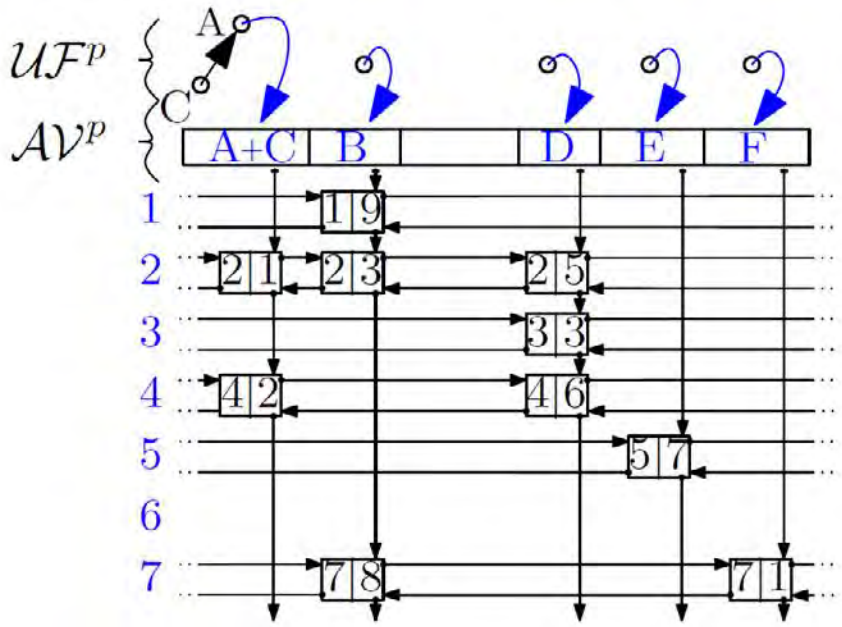
Un complejo simplicial de 10 vertices representado en un **Simplex Tree.**
El nodo más profundo representa el tetraedro del complejo. Todas las posiciones de un vértice en cierto nivel, están conectadas por una lista.

Algunos detalles :

- Una estructura que guarda en orden lexicográfico todos los simplejos del complejo.
- Utiliza de manera óptima la memoria para guardar el simplejo.
- Cualquier operación en el complejo se logra de manera óptima :
  - Agregar simplejos
  - Eliminar simplejos
  - Colapsar simplejos
  - Encontrar su frontera

GUDHI (Compressed Matrix)

	A	B	C	D	E	F
1	0	9	0	0	0	0
2	1	3	1	5	0	0
3	0	0	0	3	0	0
4	2	0	2	6	0	0
5	0	0	0	0	7	0
6	0	0	0	0	0	0
7	0	8	0	0	0	1



La **“Matriz Comprimida de Anotaciones”** de una matriz con coeficientes enteros.

Algunos detalles :

- Al ser utilizada para guardar cada paso de la reducción de la matriz cuando, siempre estamos trabajando con una matriz “Rala”.
- Optimiza la memoria utilizada para guardar la matriz.
- Las operaciones sobre la matriz para reducirse se implementan de manera óptima:
  - Sumar columnas
  - Sumar filas

Ripser (Eliminar columnas)

Idea **“Persistent Homology Computation with a Twist”[Chen, Kerber 2011]:**

- No reducir las columnas de simplejos positivos (creadores), que no sean “Escenciales”.
- Debemos reducir las columnas de la matriz $\boldsymbol{\partial_d}: \boldsymbol{C_d} \rightarrow \boldsymbol{C_{d-1}}$ en orden decreciente de dimension $\boldsymbol{d} = \boldsymbol{k+1}, \dots, \boldsymbol{1}$.
- Cuando $i = pivot\ R_j$ (en la matriz de $\boldsymbol{\partial_d}$)
  - Fijamos R_i como 0 (en la matriz para $\boldsymbol{\partial_{d-1}}$
- De igual manera obtenemos $R = D * V$ la matriz reducida, V tiene rango complete y es triangular superior.

Nota :

- Reducir columnas de **simplejos positivos** es normalmente más dificil que para los **simplejos negativos.**
- Solo tenemos que reducir las columnas de **complejos esenciales.**

Ripser (Calcular Cohomología)

Una inclusión $\boldsymbol{K} \rightarrow \boldsymbol{L}$ induce el el homomorfismo

Ripser (Comparación)

Homología : Algoritmo de estándar de reducción

$$\sum_{d=1}^{k+1} \underbrace{\binom{n}{d+1}}_{\dim C_d(K)} = \sum_{d=1}^{k+1} \underbrace{\binom{n-1}{d}}_{\dim B_{d-1}(K)} + \sum_{d=1}^{k+1} \underbrace{\binom{n-1}{d+1}}_{\dim Z_d(K)}$$

$$k = 2, n = 192: \quad 56\,050\,096 = 1\,161\,471 + 54\,888\,625$$

Homología : Utilizando eliminación

$$\sum_{d=1}^{k+1} \underbrace{\binom{n-1}{d}}_{\dim B_{d-1}(K)} + \underbrace{\binom{n-1}{k+2}}_{\dim H_{k+1}(K)} = \sum_{d=1}^{k+2} \binom{n-1}{d}$$

$$k = 2, n = 192: \quad 54\,888\,816 = 1\,161\,471 + 53\,727\,345$$

Ripser (Comparación)

CoHomología : Algoritmo de estándar de reducción

$$\sum_{d=0}^{k} \underbrace{\binom{n}{d+1}}_{\dim C^d(K)} = \sum_{d=0}^{k} \underbrace{\binom{n-1}{d+1}}_{\dim B^{d+1}(K)} + \sum_{d=0}^{k} \underbrace{\binom{n-1}{d}}_{\dim Z^d(K)}$$

$$k = 2, n = 192: \quad 1\,179\,808 = 18\,337 + 1\,161\,471$$

CoHomología : Utilizando eliminación

$$\sum_{d=0}^{k} \underbrace{\binom{n-1}{d+1}}_{\dim B^{d+1}(K)} + \underbrace{\binom{n-1}{0}}_{\dim H^0(K)} + = \sum_{d=0}^{k+1} \binom{n-1}{d}$$

$$k = 2, n = 192: \quad 1\,161\,472 = 1 + 1\,161\,471$$

Ripser (Reducción implicita)

Implementación estándar :

- La matriz de frontera $D = \partial$ para una filtración
  - Generada explicitamente y guardada en memoría
- Reducción de matriz : Guarda solo la matriz reducida R
  - Transformar **D en R** usando operaciones de columna

Reducción implicita :

- La matriz de frontera D ordenada
  - Definida implicitamente y recalculada cuando es necesario
- Reducción de matrix : Solo guardamos la matriz de coeficientes V
  - Recalcular las columnas de R = D * V cuando sea necesario
  - Generalmente, V es mucho más “Rala” que R

¿ Cuál es el rendimiento ?

गुढी GUDHI Geometry Understanding in Higher Dimensions

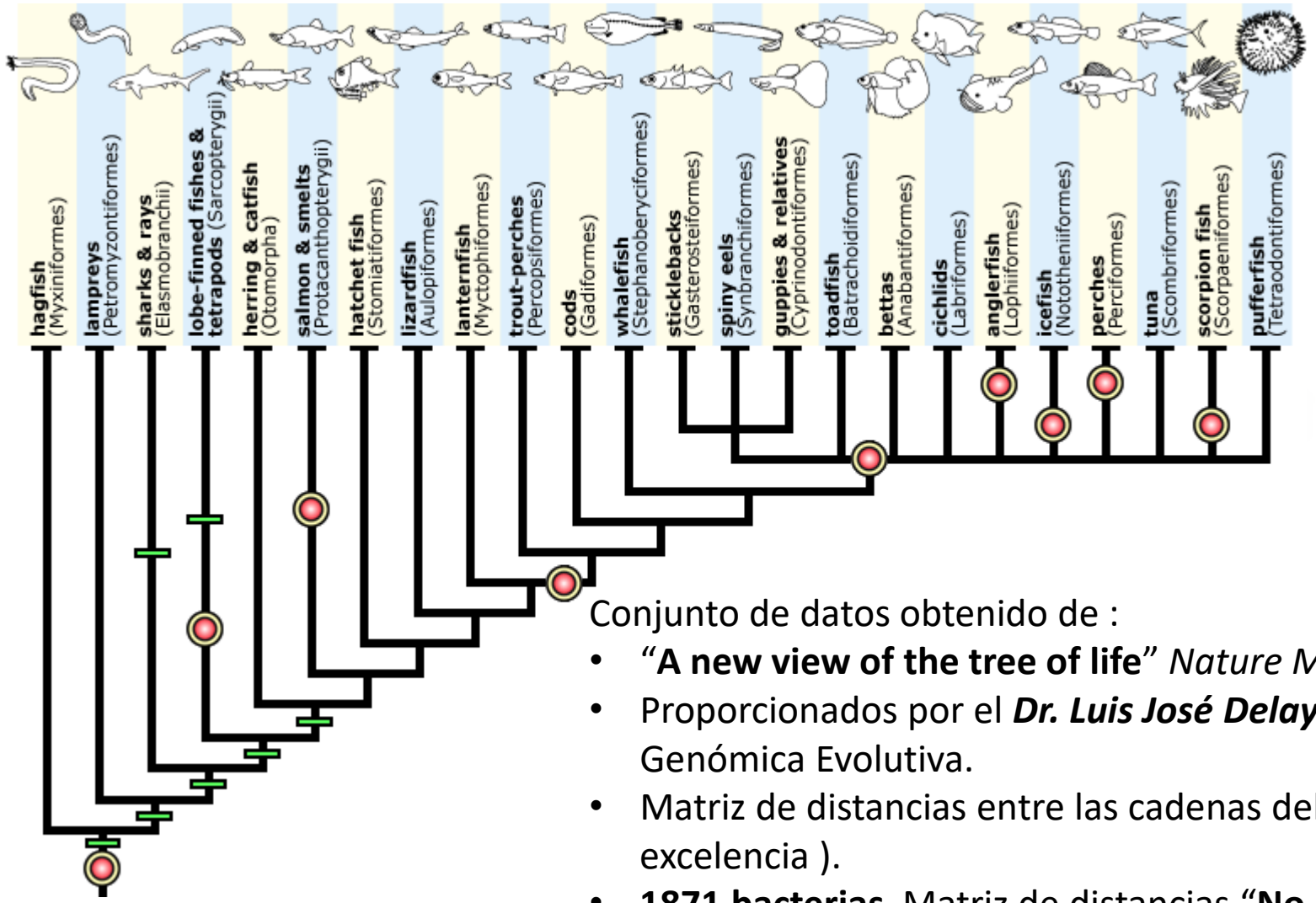
- **# Puntos : 192 en $\mathbb{S}^2$**
- **Máxima dimensión de persistencia : 2**
- **# de Simplejos hasta Dim 3 : $56\ millones$**
- **Memoria Utilizada $\approx 2.9\ GB$**
- **Tiempo de Computo $\approx 75\ segundos$**

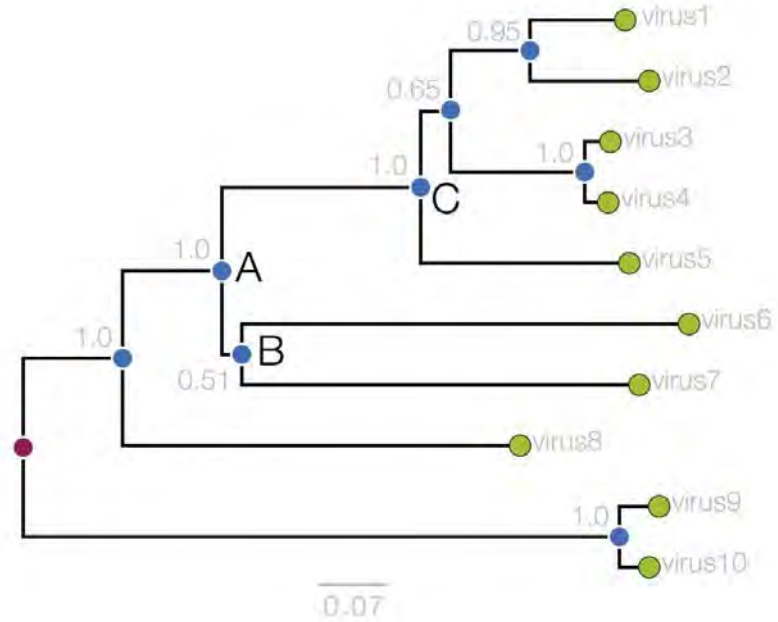
Ripser (http://ripser.org)

- **# Puntos : 192 en $\mathbb{S}^2$**
- **Máxima dimensión de persistencia : 2**
- **# de Simplejos hasta Dim 3 : $56\ millones$**
- **Memoria Utilizada $\approx 152\ MB$**
- **Tiempo de Computo $\approx 1.2\ segundos$**

Nota : Estos experimentos fueron realizados en el servidor de **TDA** de **CIMAT** usando **RStudio Server.**
Tiene **24 procesadores (Intel(R) Xeon(R) CPU E5-2643 v3 @ 3.40GHz) y 120 GB de memoría RAM.**

Analizando un conjunto de datos (**Árbol filogenético**)

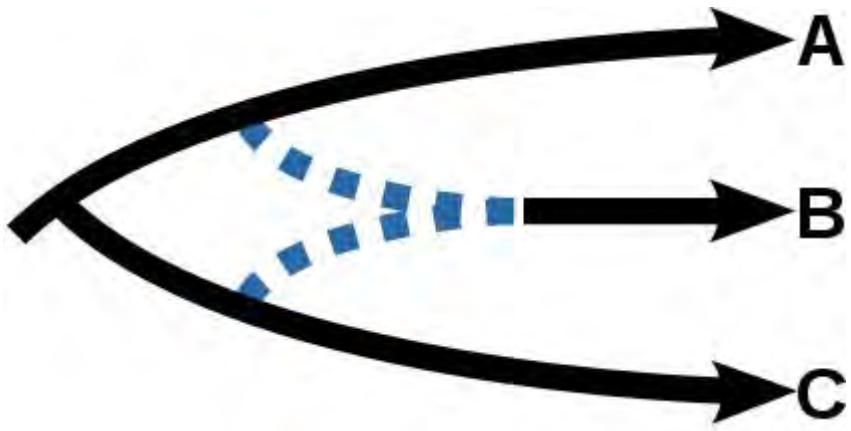




Conjunto de datos obtenido de :

- "**A new view of the tree of life**" *Nature Microbiology*, (2016)
- Proporcionados por el ***Dr. Luis José Delaye Arredondo*** , CINVESTAV, Laboratorio de Genómica Evolutiva.
- Matriz de distancias entre las cadenas del gen rRNA (el marcador evolutivo por excelencia).
- **1871 bacterias**. Matriz de distancias "**No Euclideano**". Distancia máxima **7.90841**

¿ Qué buscamos? Evolución Reticular

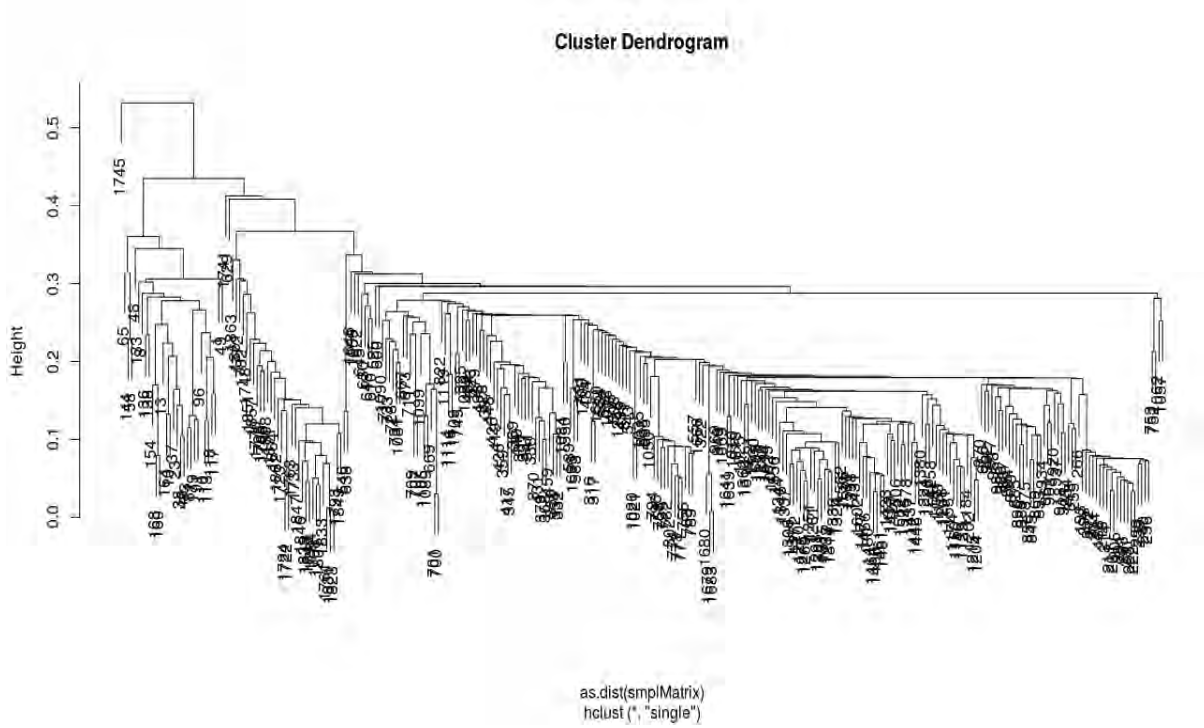


Red filogenética donde se muestra un evento de **“Evolución Reticular” :** El **linaje B** resulta de la **transferencia genética horizontal** (**TGH**) de los linajes **A y B.**

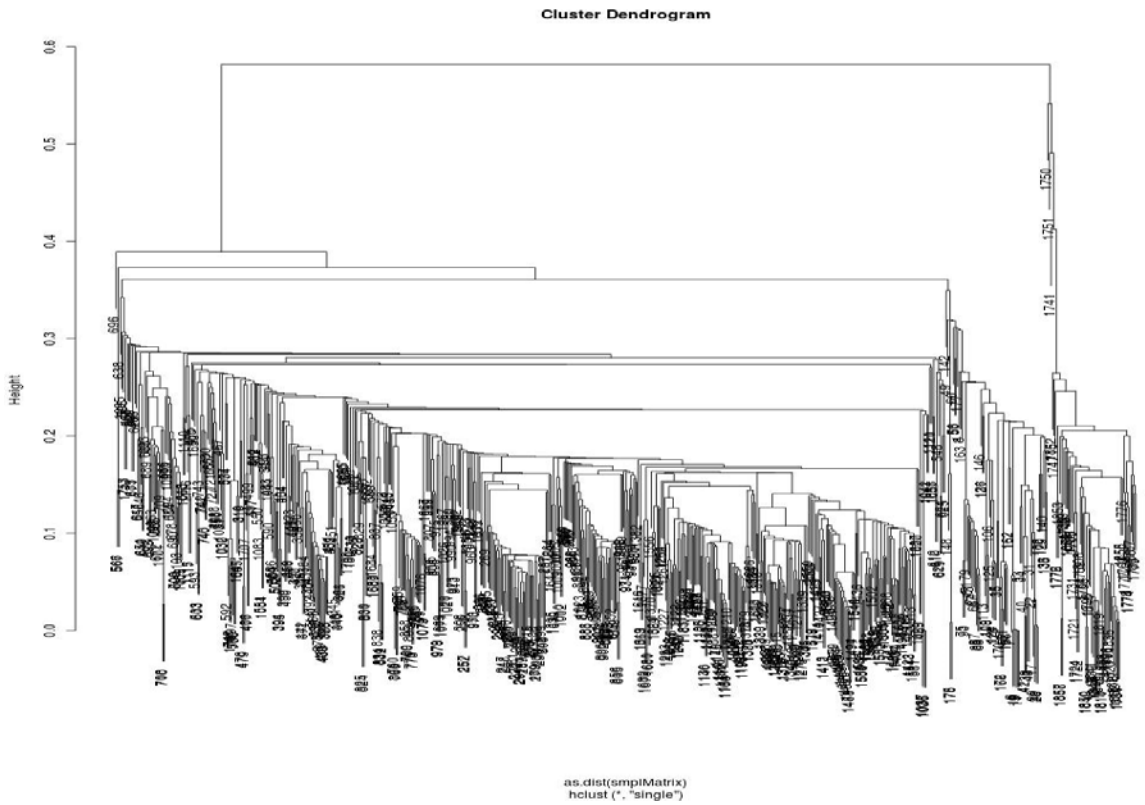
- La **Evolución Reticular** describe el origen de un “linaje” a través de la combinación parcial de linajes “ancestros”, de modo que las relaciones son mejor descritas por una **“Red Filogenética”** que un árbol que muestra solo **“Bifurcaciones”.**

- La evolución reticular ha tomado un papel clave en la evolución de algunos organismos como “Bacterias” o “Plantas de Flor”
- **[Wikipedia]**

Analizando un conjunto de datos (Dendogramas)

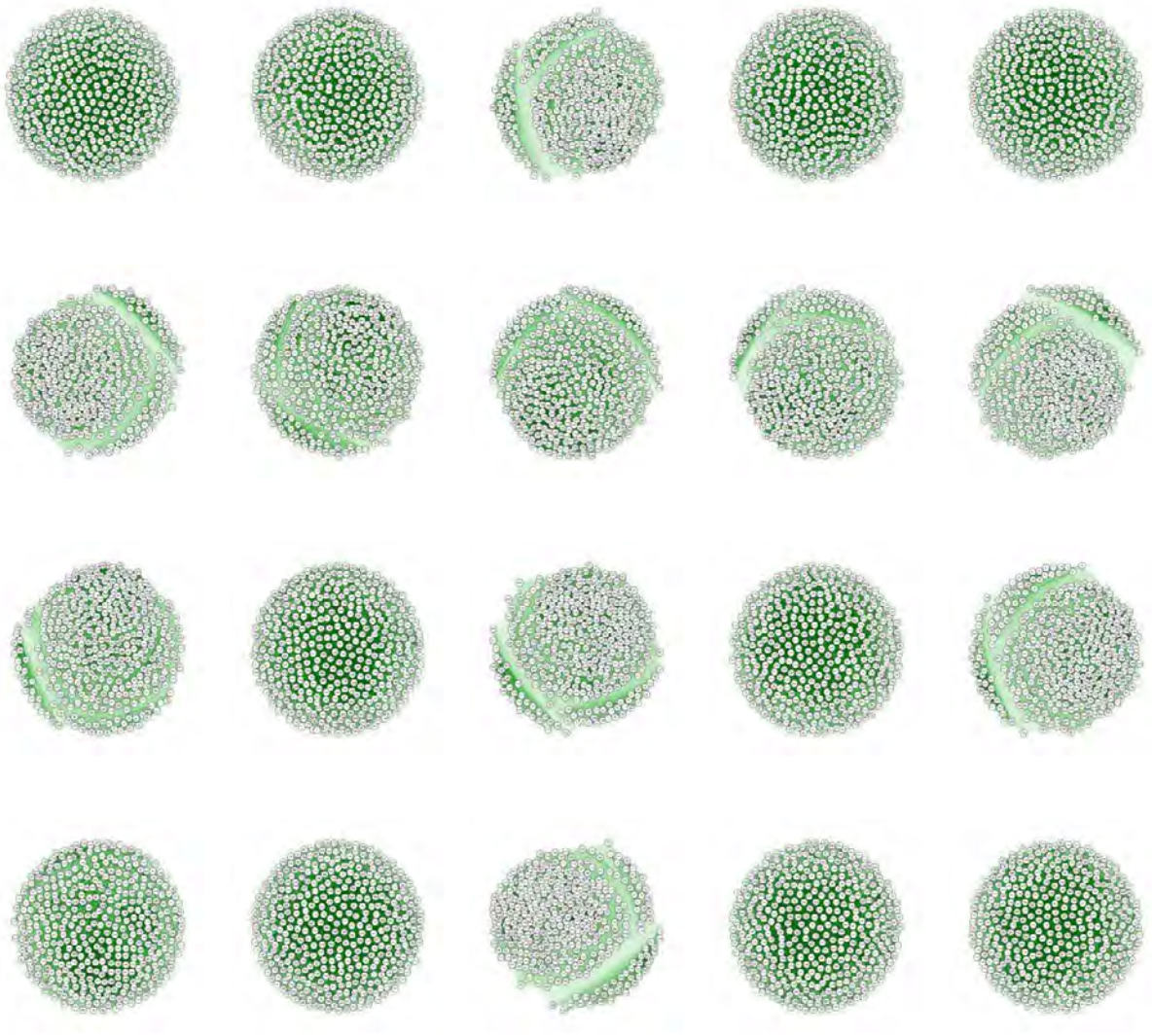


Sample Size – 600 puntos aleatorios

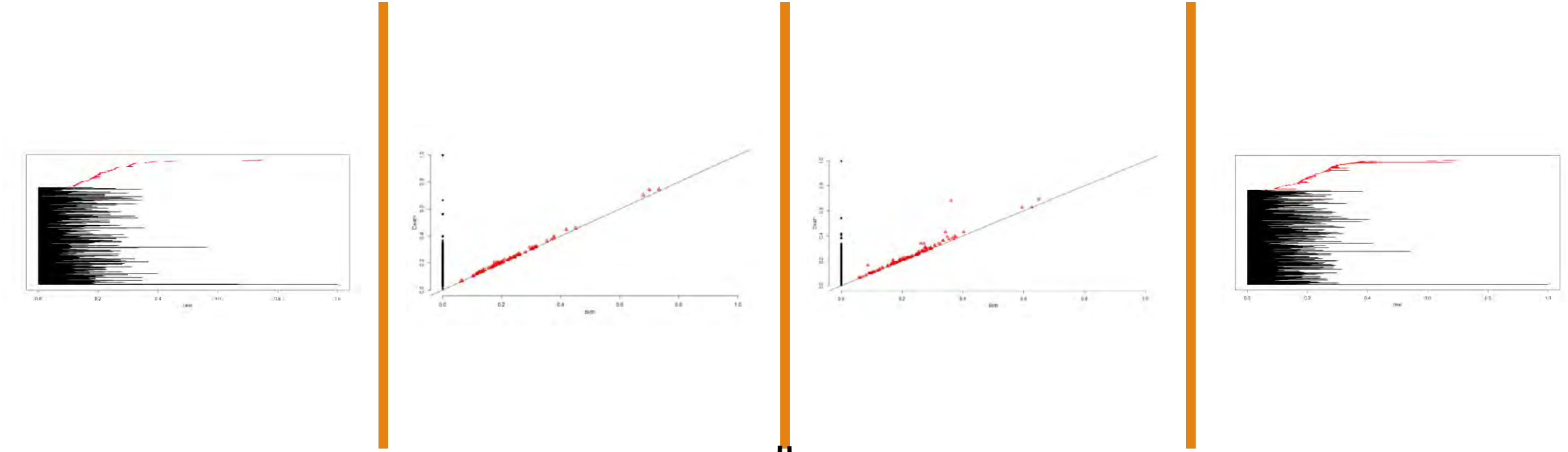


Sample Size – 700 puntos aleatorios

Analizando un conjunto de datos (Force Directed Graph)



Sample Size – 400 puntos aleatorios



Sample Size : 400 puntos – Sin ningun ciclo persistente

Sample Size : 400 puntos – Encontrando un ciclo

Analizando un conjunto de datos (Homología Persistente)

¿ Cuál es el rendimiento ?

गुढी GUDHI Geometry Understanding in Higher Dimensions

- **# Puntos : 1871**
- **Dimensión Máxima de Homología : 1**
- **Memoria Utilizada** $\approx 40\ GB$
  - ❑ El gran uso de memoria se debe a como se guardan los complejos **"Simplex Tree"**
- **Tiempo de Computo** $\approx 1\ hora$
  - ❑ Mucho del tiempo de ejecución es debido al tener que construir el complejo, la filtración y mantenerlo en memoría.

Ripser (http://ripser.org)

- **# Puntos : 1871**
- **Dimensión Máxima de Homología : 1**
- **Memoria Utilizada** $\approx 5\ GB$
  - ❑ Lo que ganamos en memoria libre lo perdemos en la libertad operaciones que podemos hacer en el complejo.
- **Tiempo de Computo** $\approx 20\ mins$
  - ❑ Debemos notar que solo podemos estudiar el conjunto de datos usando Homología Persistente con una filtración de Vietor-Rips.

Nota : Estos experimentos fueron realizados en el servidor de **TDA** de **CIMAT** usando **RStudio Server.**
Tiene **24 procesadores (Intel(R) Xeon(R) CPU E5-2643 v3 @ 3.40GHz) y 120 GB de memoría RAM.**

¿ Cómo podemos usarlo ?

RStudio Server : https://opti.cimat.mx/rstudio/

गुढी GUDHI Geometry Understanding in Higher Dimensions

- **R :** Libreria de TDA (Facil de usar)
- **Python :** GUDHI Python wrapper
- **C++ :** Versión Original **(** Un poco difícil de instalar **)**
- Muchas más herramientas para TDA además de Homología Persistente. Muy recomendable.

Ripser (http://ripser.org)

- **R : RipserOnR** https://github.com/holt0102/RipserOnR
- **Python :** Parse C++ output ejemplos en línea
- **C++ :** Facil de instalar y de usar (compilar con g++)

Nota : Estos experimentos fueron realizados en el servidor de **TDA** de **CIMAT** usando **RStudio Server.**
Tiene **24 procesadores (Intel(R) Xeon(R) CPU E5-2643 v3 @ 3.40GHz) y 120 GB de memoría RAM.**

Gracias.