Universidad de la República
Facultad de Ingeniería

Desarrollo de herramienta para visualización y análisis de evolución de virus

Memoria de proyecto presentada a la Facultad de Ingeniería de la Universidad de la República por

Federico Aicardi, Rodrigo Céspedes

en cumplimiento parcial de los requerimientos para la obtención del título de Ingeniero.

Tutor

Federico Lecumberry	Universidad de la República
María Inés Fariello	Universidad de la República
Lorena Etcheverry	Universidad de la República

Tribunal

Ernesto Dufrechou	Universidad de la República
Ewelina Bakała	Universidad de la República
Gonzalo Moratorio	Universidad de la República
Matías Di Martino	Universidad de la República/Duke University

Montevideo
jueves 25 febrero, 2021

Desarrollo de herramienta para visualización y análisis de evolución de virus, Federico Aicardi, Rodrigo Céspedes.

Esta tesis fue preparada en LaTeX usando la clase iietesis (v1.1).
Contiene un total de 107 páginas.
Compilada el jueves 25 febrero, 2021.
https://www.fing.edu.uy/

Nuestra mayor debilidad radica en renunciar. La forma más segura de tener éxito es siempre intentarlo una vez más.

Thomas A. Edison

Esta página ha sido intencionalmente dejada en blanco.

Agradecimientos

Agradecemos a nuestros tutores la dedicación y paciencia que nos brindaron durante todo el desarrollo de este trabajo.

A Diego Simón el tiempo otorgado a probar la herramienta desarrollada.

A Magalí Fernández, nuestra bióloga de cabecera.

A Leandro Marichal, por sus conocimientos en Photoshop.

Y a nuestras familias y amigos.

Esta página ha sido intencionalmente dejada en blanco.

A Magalí, Felipe, Julen y Liam

Esta página ha sido intencionalmente dejada en blanco.

Resumen

El estudio de poblaciones de virus de ARN es una tarea desafiante dada su habilidad para evolucionar rápidamente y evadir la respuesta inmune del huésped, basada en la variabilidad genética extrema.

El surgimiento de las tecnologías de NGS (Next Generation Sequencing) ha provocado que la adquisición de datos de evolución de poblaciones de virus a lo largo del tiempo sea factible. La dificultad recae en la cantidad y estructura de los datos. El análisis de datos evolutivos de virus es particularmente útil para medir la capacidad mutacional de los virus y su habilidad para adaptarse a nuevos ambientes.

A partir del trabajo "Análisis y Visualización de la Evolución de Virus" [31] en este proyecto se desarrolla una herramienta para analizar y visualizar los datos obtenidos por experimentos evolutivos de distintas especies de virus de ARN, con el fin de poder entender los mecanismos que subyacen la adaptación de estos virus a distintos ambientes.

Se desarrollaron dos nuevas aproximaciones metodológicas, una con una componente mas visual y otra más analítica.

Como resultado se obtuvo una herramienta que integra todos estas aproximaciones brindando una interfaz al usuario. Se realizaron pruebas con un conjunto de datos conocidos, logrando obtener información de interés ya relevada.

Esta página ha sido intencionalmente dejada en blanco.

Tabla de contenidos

Agradecimientos III

Resumen VII

1. Introducción **1**
1.1. Descripción del proyecto 1
1.2. Objetivos 1
1.3. Alcance 2
1.4. Organización del documento 2

2. Genética, evolución, virus y modelo conceptual **3**
2.1. Material genético 3
2.2. Evolución 5
2.2.1. Deriva genética, efectos fundador y cuello de botella 6
2.2.2. Mutación 6
2.2.3. Selección natural 7
2.3. Virus 8
2.3.1. Replicación viral 10
2.3.2. Coxsackievirus 12
2.4. Deep sequencing - NGS 13
2.5. Descripción de experimento y conceptos relacionados 13
2.5.1. Modelado de la realidad del proyecto 14

3. Análisis de los datos y resultados previos **17**
3.1. Análisis mediante variación de frecuencia de codones dominantes . 20
3.2. Definición de un espacio de variaciones 21
3.3. Análisis estadístico de los máximos de la entropía 23
3.4. Detección de codones representativos de las cepas mediante representación de la evolución viral en baja dimensión 25

4. Visualización mediante t-SNE **27**
4.1. Algoritmo t-SNE 27
4.2. t-SNE con Scikit-learn 30
4.3. Aplicación de t-SNE a los datos 31

Tabla de contenidos

5. Agrupación y descomposición de series temporales **35**
5.1. Agrupación de series temporales 35
5.2. Dynamic Time Warping 36
5.3. Descomposición de la serie temporal - Singular Spectrum Analysis 37
5.4. Aplicación del enfoque de series temporales a los datos 39
5.4.1. Agrupaciones con las trayectorias de entropía 40
5.4.2. Agrupación con Singular Spectrum Analysis 42

6. Diseño y desarrollo de la herramienta **47**
6.1. Fortalezas y debilidades de la solución existente 47
6.1.1. Fortalezas 47
6.1.2. Debilidades 47
6.2. Requerimientos de la herramienta a desarrollar 48
6.3. Arquitectura general de la solución 49
6.3.1. Capa de presentación 50
6.3.2. Capa de negocio 52
6.3.3. Capa de persistencia 65
6.4. Distribución de la Herramienta 72
6.4.1. GitHub 72
6.4.2. PyPI 72

7. Conclusiones y trabajo futuro **73**
7.1. Conclusiones 73
7.2. Trabajo futuro 73

A. Anexo Herramientas Utilizadas **75**
A.1. PyQT 75
A.1.1. QtDesigner 76
A.2. Numpy 77
A.3. Scikit-learn 77
A.4. Matplotlib 78
A.5. Scipy 79
A.6. Pillow 80

Referencias **85**

Índice de tablas **89**

Índice de figuras **90**

Capítulo 1

Introducción

1.1. Descripción del proyecto

Las enfermedades infecciosas emergentes representan una amenaza permanente debido a nuevas epidemias en regiones endémicas y a brotes importados en otros países. Los agentes causantes de enfermedades como Ébola, Chikunguya, Zika o incluso el COVID-19 el cual estamos hoy en día padeciendo son virus de ARN. Su habilidad para evolucionar rápidamente y evadir la respuesta inmune del huésped, basada en la variabilidad genética extrema, puede explicar la propagación y devastación causada por estos virus [31].

En el trabajo "Análisis y Visualización de la Evolución de Virus" [31], se analizan datos que provienen de tres cepas de virus para los cuales se realizan tres experimentos donde se releva su genoma en veinte pasajes (cada seis generaciones). En el mismo, se analizó el uso de la entropía de Shannon para modelar matemáticamente la variabilidad en el ARN del virus y lograr visualizar su evolución para detectar posiciones de interés en la cadena.

A partir de las aproximaciones metodológicas para analizar los datos desarrolladas en el proyecto mencionado anteriormente [31], éste proyecto consiste en integrar éstas funcionalidades en una única herramienta o entorno de análisis que pueda ser utilizado con datos de virus de ARN genéricos. Además se desea agregar herramientas a las ya existentes de análisis y/o visualización de la evolución de virus de forma de poder encontrar posiciones de interés dentro de la cadena genómica para su posterior estudio por el usuario.

1.2. Objetivos

Nucleando las herramientas desarrolladas en el trabajo previo, extrayendo y analizando información, se busca detectar posibles posiciones de interés desde el punto de vista adaptativo del virus, dentro de la cadena de ARN para el posterior estudio por el investigador. Se pretende desarrollar una herramienta que permita a los investigadores estudiar experimentos de evolución de virus de ARN agregando

valor a los mismos al proveer una capa de visualización y análisis. Para alcanzar este objetivo, se plantean los siguientes objetivos específicos:

1. Estudio del trabajo previo
    - Estudio de la temática para entendimiento de los datos analizados
    - Comprensión de los análisis realizados
2. Estudio de nuevos métodos de análisis y visualización
3. Implementación de la herramienta

1.3. Alcance

El alcance del presente proyecto comprende:

- Diseño y desarrollo de una herramienta que proporcione los métodos realizados en el trabajo "Análisis y Visualización de la Evolución de Virus" [31], para que pueda ser utilizado con datos genéricos de experimentos sobre virus de ARN permitiendo el análisis de los mismos.
- Estudio e implementación de nuevos métodos de análisis de datos de evolución viral.

No se encuentra dentro del alcance del proyecto el formatear los datos de entrada, se asumen que los datos a ingresar, tienen un formato ya establecido.

1.4. Organización del documento

Este documento está organizado de la siguiente manera:

- En el capítulo 2 se tratan algunos conceptos del dominio, en particular sobre genética, virus y evolución. Además se brinda una descripción de los datos que se analizan y el modelo conceptual correspondiente.
- En el capítulo 3 se analiza el proyecto anterior, desde los métodos teóricos, hasta los resultados obtenidos.
- En los capítulos 4 y 5 se exponen las nuevas aproximaciones metodológicas. Se explican tanto desde el punto de vista teórico, así como también los resultados obtenidos.
- En el capítulo 6 se presenta el diseño y el desarrollo de la herramienta implementada.
- En el capítulo 7 se detallan las conclusiones y las posibles lineas de trabajo a futuro.

Se brinda un anexo donde se profundiza en las bibliotecas utilizadas en la implementación del proyecto. Además, se brinda un glosario de términos.

Capítulo 2

Genética, evolución, virus y modelo conceptual

Con el objetivo de que este documento quede auto contenido, se tratarán algunos conceptos relacionados que ayudan a entender la temática. Comenzaremos con la noción de material genético. Luego se introducen el concepto de evolución junto con los mecanismos que llevan al proceso evolutivo. En la sección 2.3 se reseña cómo es la estructura de la cadena genómica de los virus ARN y el proceso de replicación de los virus. En la sección 2.4 se da una breve explicación de la técnica que permite la adquisición de los datos estudiados. Finalmente, en la sección 2.5 se introducen los conceptos de experimento y pasaje para entender la estructura de los datos analizados y se presenta el modelo utilizado para el desarrollo de la herramienta.

2.1. Material genético

El material genético constituye la manera en que se guarda y transmite la herencia biológica de generación en generación. Para este fin, la mayoría de los organismos vivos, utilizan el ácido desoxirrobouncleico (ADN). Sin embargo, algunos genomas de virus como los estudiados en esta tesis, utilizan el ácido ribonucleico (ARN). La cristalografía de rayos X, ha proporcionado la mayor parte de la información que conocemos sobre la estructura del los ácidos nucleicos. Éstos se componen por unidades individuales repetidas denominadas nucleótidos. Cada unidad comprende tres subunidades unidas entre sí: un grupo fosfato, un azúcar y una de las cuatro bases. Las bases nitrogenadas, son moléculas heterocíclicas aromáticas planas y se dividen en dos grupos: las pirimidínicas, timina y citosina y las purínicas, adenina y guanina. En el caso del ARN, la timina se reemplaza por uracilo [36].

Las unidades de nucleósidos individuales se componen del azúcar y la base nitrogenada, y se unen en un ácido nucleico de manera lineal, a través de grupos fosfato. Por tanto, la unidad de repetición completa en un ácido nucleico es un nucleótido. Las secuencias de ácidos nucleicos y oligonucleótidos utilizan códigos

de una sola letra para los nucleótidos: A, T, G, C y U. En la figura 2.1 se puede ver la estructura del ADN y ARN, con su respectivos nucleótidos.

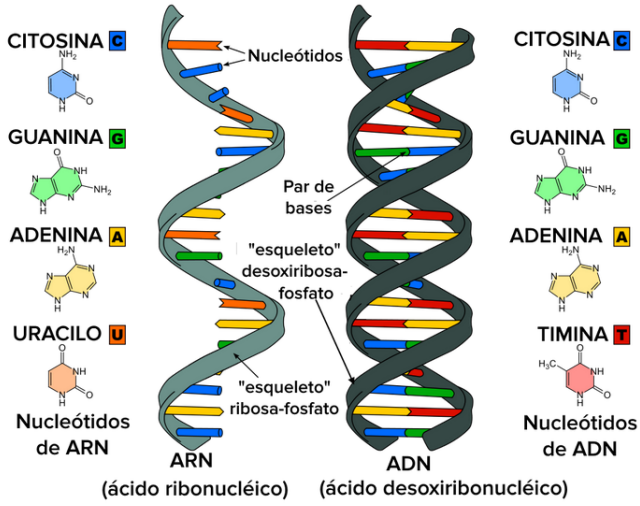


Figura 2.1: Ácidos nucleicos. A la izquierda de la figura ARN. A la derecha ADN [1].

Cada hélice está formada por nucléotidos unidos por enlaces fosfodiéster, en el que participa un grupo fosfato y el azúcar. Las dos hélices se mantienen unidas por puentes de hidrogeno, donde dos átomos electronegativos comparten un protón [12]. El ARN, por su parte, es capaz de formar una gran variedad de estructuras. Sin embargo, las secuencias apropiadas forman fácilmente estructuras de ARN de doble hebra antiparalelas, análogas al ADN dúplex. El uracilo participa en pares de bases U-A que son completamente isomorfos con los A-T en el ADN dúplex [36].

El gen es la unidad fundamental, física y funcional, de la herencia, que transmite la información de una generación a la siguiente. Cada gen, o secuencia de interés, ocupa una región específica en la molécula de ácido nucleico, denominada locus. En una población, en cada locus pueden existir diversas secuencias alternativas, las cuales se denominan alelos [12].

Tanto el ADN como el ARN, son luego traducidos en proteínas cuyas unidades básicas son los aminoácidos. Todos los organismos vivos están construidos a partir de veinte aminoácidos. Estos son codificados mediante tripletes de nucleótidos adyacentes denominados codones. De las sesenta y cuatro posibles combinaciones, sesenta y uno son codificadas a aminoácidos mientras que las restantes tres son codones de parada (marcadores) [50]. En definitiva, se puede ver la cadena de ARN (y por lo tanto la información hereditaria) como una secuencia ordenada de codones.

El código genético se describe como degenerado o redundante, porque un solo aminoácido puede estar codificado por más de un codón. También es importante señalar que el código genético no se superpone, lo que significa que cada nucleótido es parte de un solo codón; un solo nucleótido no puede ser parte de dos codones adyacentes. Además, el código genético es universal, esto quiere decir que el mismo

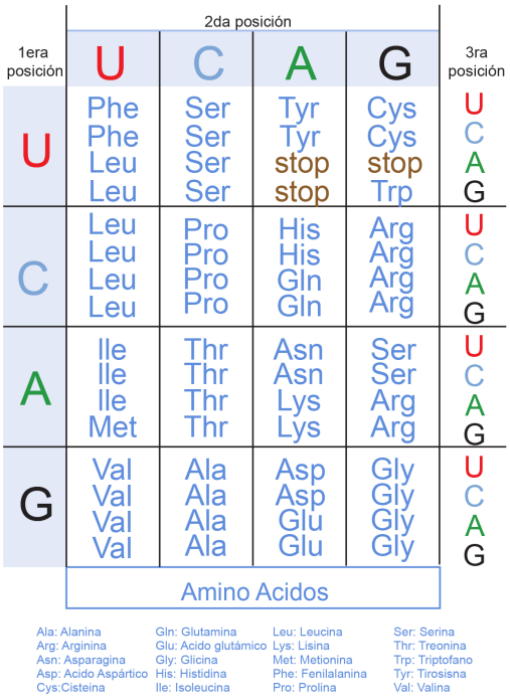


Figura 2.2: Código genético en el ARN [18].

triplete en diferentes especies codifica para el mismo aminoácido. En la figura 2.2 se puede observar como se interpreta el mismo.

2.2. Evolución

Se puede decir que la evolución es esencialmente un proceso de cambio genético en el tiempo. Este proceso transcurre a través de las variaciones heredables que pasan de una generación a otra de organismos, en poblaciones naturales. La genética de poblaciones se refiere a la variación en la transmisión de la información genética de una generación a la siguiente. Ésta variación que ocurre generación tras generación, se denomina microevolución, constituyendo el estudio de procesos evolutivos en escalas cortas de tiempo. Por este motivo, la unidad básica de tiempo en este contexto es una generación. Por tanto, la evolución se puede ver como un cambio acumulativo de las porciones de las diferentes variantes de los genes en las poblaciones, donde la variación genética surge por mutaciones que modifican el material genético pre-existente, y la selección natural dicta el destino de tales variantes. Los agentes que cambian las frecuencias alélicas de las poblaciones, es decir los factores de evolución, son la mutación, la deriva genética, la migración y la selección natural [10].

2.2.1. Deriva genética, efectos fundador y cuello de botella

Considerando una población de tamaño finito, donde no actúa la selección, todos los alelos en una población tienen igual probabilidad de dejar descendientes. Esto hace posible que la composición genética de la población vaya cambiando a lo largo del tiempo. Se puede observar en la práctica que la descendencia que cada uno deje en la siguiente generación será algo meramente azaroso. De acuerdo con ello, en cada generación se espera una fluctuación al azar de las frecuencias alélicas en las poblaciones. Puede suceder que en algún momento, un tipo de los alelos no se transmita a la siguiente generación, y por lo tanto se habrá perdido para siempre. El proceso de fluctuación no continúa indefinidamente, sino que se detiene en una de dos situaciones: cuando el alelo se fija (es decir, llega a ser el 100 % de la población) o cuando se extingue (como se mencionaba anteriormente). Por estos motivos se puede decir que la deriva suele implicar pérdida de variabilidad genética, y por ende contrarresta la entrada de variabilidad genética por mutaciones [10] [32].

Asociado al concepto de deriva, se puede dar lo que se denomina **cuello de botella**. Cuando una población se ve notoriamente disminuida al menos por una generación, se dice que se ve afectada por el efecto cuello de botella. Debido a la disminución de población, la variabilidad genética también se verá decrementada. Esto implica que la población haya perdido variabilidad y pueda no adaptarse a presiones selectivas.

Por otra parte, se denomina **efecto fundador** cuando unos pocos individuos de una población establecen una nueva colonia. Ésta nueva población, dado que se trata de una cantidad pequeña, tendrá menos variabilidad genética que la población original. Por ejemplo en la nueva colonia se pueden fijar alelos que eran raros en la población original, si los individuos fundadores poseen dichos alelos. A su vez, alelos comunes en la población original se pueden perder en la colonia por el mismo motivo.

2.2.2. Mutación

La variación es la materia prima de la evolución, y sin ella ésta no es posible. Cual será el destino de dicha variación, se define en el plano de la genética de población, que incluye, el proceso de la selección natural, pero también la deriva genética. Una mutación es en definitiva un cambio estable y heredable en el material genético. Las mutaciones alteran el material genético (la secuencia de ADN o ARN) introduciendo nuevas variantes y por tanto aumentando la diversidad genética. Muchas de éstas serán eliminadas, pero eventualmente algunas pueden tener éxito.

La tasa de mutación de un gen o una secuencia de ADN (o ARN) es la frecuencia en la que se producen nuevas mutaciones en ese gen o secuencia en cada generación. Una alta tasa de mutación se puede ver como un mayor potencial de adaptación a un cambio ambiental ya que permite que más variantes genéticas sean posibles, aumentando la probabilidad de obtener la variante adecuada. De igual manera, una alta tasa de mutación aumenta el número de mutaciones perjudiciales o deletéreas, haciendo a estos individuos menos adaptados. Las mutaciones

son cambios al azar y no tienen ninguna dirección respecto a la adaptación. Estos cambios representan la tasa de mutación, la cual es modulada por la selección natural para poder enfrentarse a los compromisos contrapuestos de estabilidad-cambio impuestos por el ambiente [10].

2.2.3. Selección natural

De manera muy esquemática, la idea básica de la selección natural es que los organismos con rasgos heredables que les ayudan a sobrevivir y reproducirse en el ambiente donde habitan, dejarán más descendientes que los organismos que carecen de dichas características. Dado que dichos rasgos son heredables, pasarán a su descendencia la cual también tendrá ventaja en la supervivencia y reproducción. Por este motivo, a medida que pasen las generaciones, la supervivencia y reproducción diferenciales llevarán a un aumento progresivo en la frecuencia de las características que así lo permiten dentro de la población.

Considerando todos los mecanismos de la evolución, la selección natural es el único que, de manera consistente, adapta a las poblaciones o las hace más adecuadas para vivir en su medio ambiente con el paso del tiempo.

La selección natural es un proceso que ocurre cuando se cumplen las siguientes condiciones necesarias y suficientes:

- Existe variación fenotípica entre individuos de una población. Cuando se habla de variación fenotípica se hace referencia a las propiedades reales observadas de los individuos, como pueden ser la morfología, el desarrollo o el comportamiento. Vale la pena aclarar que éstas características observadas no tienen por qué ser visibles, por ejemplo la presencia de una enzima.

- Esta variación es al menos en parte heredable (es decir se transmite en los genes).

- Dicha variación afecta el éxito reproductivo de los individuos (existe una correlación entre la variación y el éxito reproductivo)

Se trata de un proceso poblacional que puede operar en cada generación, pero a su vez, puede requerir un gran número de ellas para manifestar su eficacia. Además, no hay garantías de poder constatar la ocurrencia en un período corto de tiempo.

Se puede afirmar que la selección natural es un proceso probabilístico. Involucra a una cierta característica fenotípica, heredable al menos en parte, cuya manifestación aumenta o reduce la probabilidad de contribución reproductiva de los individuos a las siguientes generaciones. Donde además existe una relación consistente, entre la variación fenotípica (y, por su correlación con ella, la variación genotípica) y el éxito reproductivo. Dicha relación no tiene porqué involucrar diferencias dramáticas entre los individuos, o muy fuertes correlaciones entre fenotipo y éxito reproductivo.

Éste es un proceso complejo en doble medida. Por un lado se sabe que la determinación de la relación entre genotipo y fenotipo puede tomar muchas formas.

A la complejidad de la genética, se le debe sumar la determinación de la relación entre fenotipos y éxito reproductivo. La selección no implica que el mayor éxito reproductivo esté directamente vinculado a procesos íntimamente involucrados en la reproducción (como pueden ser una mayor producción primaria de gametos o una estrategia de inversión permanente en la reproducción). Además, tampoco implica que exista un único fenotipo favorable en una población.

Es un proceso que toma muchas formas, ya que si bien tomar en cuenta el ambiente a la hora de valorar la capacidad adaptativa del fenotipo parece bastante evidente, se debe tener en cuenta también que dicho valor adaptativo depende también del contexto genético en que aparece. Una característica fenotípica no tiene signo adaptativo en sí misma sino en relación a cómo influye en el éxito de supervivencia y reproductivo de los individuos, y en relación con sus restantes características [32].

En ausencia de los previos mecanismos, y considerando una población de tamaño infinito, la composición genética de la población se mantiene en equilibrio. Este equilibrio es denominado Equilibrio de Hardy-Weinberg y es utilizado como Hipótesis nula para estudiar si sobre un alelo está actuando algún mecanismo evolutivo y no meramente el azar.

2.3. Virus

Las partículas virales se diferencian de otros microorganismos en su organización y composición:

- Tienen un sólo tipo de ácido nucleico, ARN o ADN.
- Poseen una cubierta proteica (cápside) que encierra el ácido nucleico y algunos están rodeados por una envoltura lipídica de origen celular.
- Carecen de sistemas enzimáticos propios.

Esta última característica hace que los virus requieran necesariamente de la maquinaria enzimática de las células huésped para sintetizar sus proteínas y reproducirse y, por lo tanto, son parásitos obligados. Además, son incapaces de generar o almacenar energía en forma de trifosfato de adenosina (ATP), por lo que la obtienen de la célula huésped. De manera similar, la célula huésped brinda los elementos básicos para la síntesis de sus componentes, como aminoácidos, nucleótidos y lípidos (grasas) [14]. A diferencia de las células, los virus no aumentan de tamaño para dividirse y multiplicarse. En la replicación viral, la partícula se desintegra y luego se sintetiza cada uno de los componentes que se ensamblan ordenadamente para formar las nuevas partículas virales. De esta manera, se producen réplicas del virus progenitor [5]. Existe gran variedad de tamaño y forma de los virus los cuales van desde 28 nm en los virus más pequeños hasta 300 nm en los virus más grandes que se conocen. Los virus más simples se componen por un solo tipo de ácido nucleico (ADN o ARN) rodeado de una envoltura proteica, la cápside. Las

subunidades proteicas codificadas por el genoma viral, se ensamblan según principios geométricos y se forman las simetrías icosaédrica o helicoidal. En los virus más complejos, o que tienen envoltura lipídica, también se observan formas esféricas, filamentosa o pleomórficas. La estructura básica de ácido nucleico y cápside recibe el nombre de nucleocápside y constituye en los virus desnudos la partícula viral completa o virus. El término virión refiere a las partículas virales o virus potencialmente infecciosas [5]. En la figura 2.3 se pueden ver ejemplos de diferentes formas geométricas de viriones.

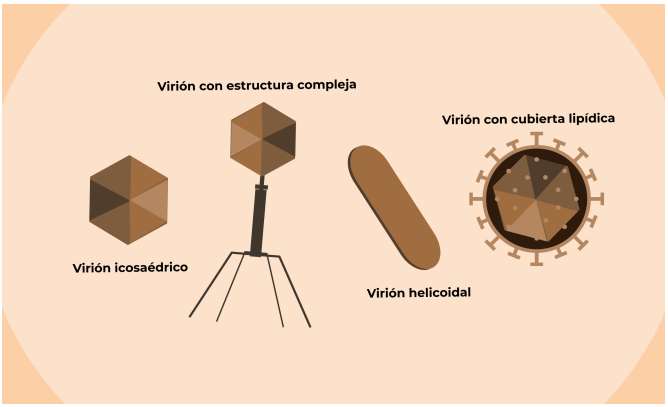


Figura 2.3: Ejemplo de formas geométricas de viriones [5].

Cápside

La cápside está compuesta por proteínas organizadas en subunidades morfológicas, conocidas como capsómeros. La forma de distribución de los capsómeros así como el número de ellos depende de cada tipo de virus. La cápside tiene tres funciones:

- Protege el ácido nucleico de la digestión por enzimas.
- Contiene sitios especiales en su superficie que permiten que el virión se adhiera a una célula huésped.
- Poporciona proteínas que permiten que el virión penetre en el huésped. membrana celular y, en algunos casos, para inyectar el ácido nucleico infeccioso en el citoplasma de la célula.

Envoltura

Muchos tipos de virus tienen una envoltura de glucoproteína que rodea la nucleocápside. La envoltura está compuesta por dos capas de lípidos intercaladas con moléculas de proteínas (bicapa de lipoproteínas). El virus obtiene las moléculas de lípidos de la membrana celular durante el proceso de gemación viral. Sin embargo, el virus reemplaza las proteínas de la membrana celular con sus propias proteínas, creando una estructura híbrida de lípidos derivados de células y proteínas derivadas de virus. Muchos virus también incorporaran glicoproteínas en sus envolturas

(espículas o glicoproteínas de superficie) que tienen un importante papel en el reconocimiento de receptores específicos de la superficie celular, el paso inicial de adherencia a la célula huésped para la multiplicación viral.

Ácido nucleico

Al igual que en las células, el ácido nucleico contiene la información genética para la síntesis de todas las proteínas. Una partícula viral tiene un solo tipo de ácido nucleico (ADN o ARN). En particular, hay dos tipos de virus basados en ARN: de hebra positiva y de hebra negativa. En el primer caso, el genoma actúa como ARN mensajero para la síntesis (traducción) directa de la proteína viral, mientras que en el segundo, el virión debe contener una enzima, llamada ARN polimerasa dependiente de ARN (transcriptasa), que cataliza la producción del ARN mensajero complementario a partir del ARN genómico del virión, para que pueda ocurrir la síntesis de proteína viral. Finalmente, el ácido nucleico de los virus se presenta en doble o simple cadena, segmentado o no, lineal o circular. Las características estructurales y del ácido nucleico, se combinan en diferentes maneras para determinar varios grupos de virus [5].

2.3.1. Replicación viral

Aunque el ciclo de vida replicativo de los virus difiere mucho entre las diferentes categorías de virus, implica tres amplias etapas que están presentes en todos los virus: inicio de la infección, replicación y expresión del genoma y liberación de viriones maduros desde la célula infectada hacia el espacio extracelular. A un nivel más detallado, la replicación viral se puede dividir en seis etapas: unión a la célula huésped, entrada o penetración de la partícula, liberación del material genético (descubrimiento), transcripción y replicación del genoma, ensamblaje de las partículas hijas, maduración y liberación al medio extracelular. Las diferencias entre los procesos de replicación de diferentes virus, obedecen a la biología de la célula huésped y la naturaleza del genoma del virus. En la figura 2.4 se muestra un esquema de la replicación viral.

En la primer etapa de la replicación viral, el virus debe anclarse a la célula huésped. En dicha etapa son importantes la interacción de sitios específicos de la partícula viral con receptores celulares específicos. Por este motivo, en ésta etapa juega un rol preponderante la membrana en el caso de los virus que la poseen o la cápside en aquellos que carecen de ella.

La etapa de penetración, una vez unidos, se puede dar de varias formas. La más común es la viropexis, donde el virus queda englobado en una vesícula dentro del citoplasma celular (es el caso de la figura 2.4). Otra forma en que el virus puede ingresar a la célula son la penetración donde el virus simplemente cruza la membrana plasmática y la partícula viral queda directamente incluida en el citoplasma. Por último, la penetración se puede dar por fusión de la envoltura viral con la membrana plasmática.

En la etapa de liberación del material genético, el virus se desintegra, dejando libre su ácido nucleico. De esta manera comanda su propia replicación, además de

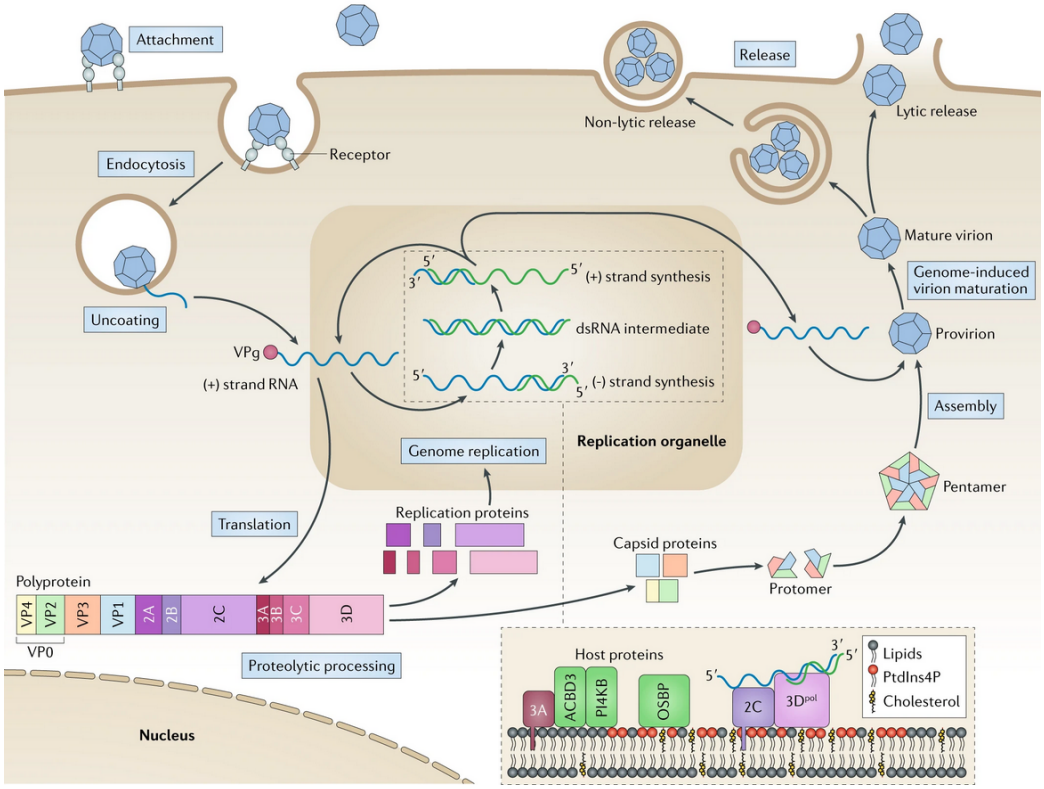


Figura 2.4: Esquema de replicación viral y sus etapas. Unión o *Attachment*. Penetración (en éste caso por viropexis o *endocytosis*). Liberación del material genético (*Uncoating*). Transcripción y replicación del genoma viral (*Translation* y *Genome replication*). Ensamblaje de las partículas hijas (*Assembly*). Maduración y liberación al medio extracelular (*Genome-induced virion maturation* y *Release*) [8].

la síntesis de las proteínas necesarias para integrar nuevas partículas. Cada grupo viral pierde la cápside y su envoltura (si la posee) de una manera característica.

La etapa de transcripción y replicación es un fenómeno muy heterogéneo. Sin embargo, de manera general se puede decir que siempre el genoma viral es el elemento capaz de gobernar su autorreplicación y de trasmitir la información estructural y funcional a la descendencia resultante de una infección. Además, en esta etapa también se diferencian dos conjuntos de genes, los precoces y los tardíos. Los primeros se encargan de codificar proteínas necesarias para la copia de la molécula de ácido nucleico. Los segundos, de codificar las proteínas estructurales y las proteínas para el ensamblaje.

Para los virus desnudos (es decir aquellos que no poseen envoltura) el ensamblaje y maduración consiste en la unión de los capsómeros para formar la cápside y la posterior unión de ésta con el genoma. Para estos casos, la liberación depende mucho tanto del virus como de las células huésped. En los virus con envoltura, luego de formarse la cápside (el cual es un proceso más complejo que en el caso de los virus sin envoltura), el virus debe rodearse de la envoltura. Posteriormente la

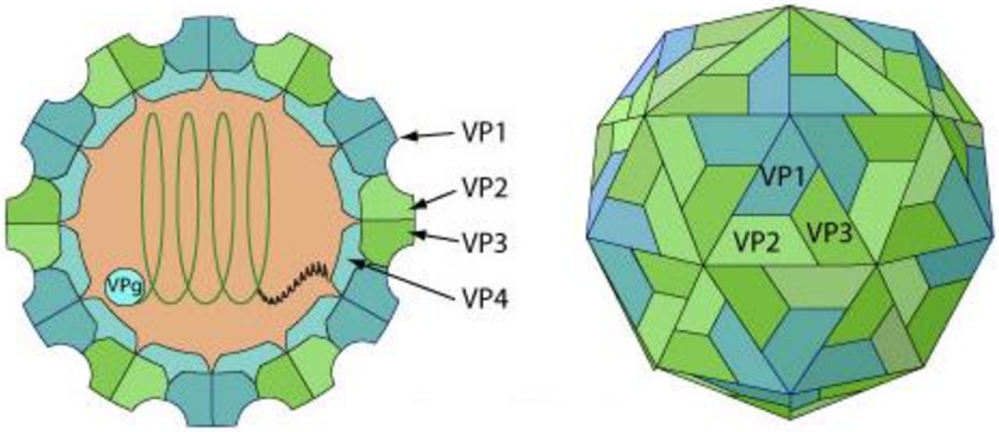


Figura 2.5: Virión de la familia Picornavirus a la que pertenece el Coxsackievirus [58].

partícula se aproxima a la membrana plasmática y se produce la evaginación de la membrana y luego el desprendimiento del brote [5].

2.3.2. Coxsackievirus

Dado que los datos analizados corresponden a la especie coxsakievirus, en esta sección se da una breve introducción a este virus.

Los coxsackievirus pertenecen al género Enterovirus, de la familia Picornaviridae. Al igual que otros enterovirus, los coxsackievirus del grupo A y B (CVB) (enterovirus que se replican a 37 °C) tienden a infectar órganos internos. El virión consiste en una partícula icosaédrica no envuelta, de aproximadamente 30 nm de diámetro. La cápside está compuesta por cuatro proteínas, VP1, VP2, VP3 y VP4. Una molécula de VP1, VP2, VP3 y VP4 forma un protómero. Cinco protómeros componen un pentámero y doce pentámeros forman la cápside, compuesta por sesenta protómeros. Mientras que VP1, VP2 y VP3 aparecen en la capa exterior de la cápside, VP4 es una proteína interna [7]. Estas cuatro proteínas quedan determinadas por los codones presentes en la cadena genómica del virus. En la figura 2.5 se puede ver el virión de la familia del Picornavirus a la cual pertenece el Coxsackievirus [7].

La información genética de CVB3 se almacena en un pequeño ssRNA (*single strained RNA* o ARN simple hebra) de $7,4$ Kb de longitud. El genoma es una molécula lineal infecciosa debido a su polaridad positiva. El ssRNA codifica cuatro proteínas estructurales VP1 a VP4 y siete proteínas no estructurales. Las proteínas estructurales están destinadas a ensamblarse juntas de la manera correcta, para formar la cápside del virus, mientras que las proteínas no estructurales tienen el objetivo sinérgico de asegurar la producción de ARN viral por la célula huésped y permitir la liberación de partículas CVB3 al espacio extracelular [7].

Es conocido que varios órganos pueden verse afectados tras la infección del género Coxsackievirus. Por ejemplo, el intestino, el corazón, el páncreas y el sistema

neuronal. Esto puede causar daños graves o incluso la muerte. Particularmente se ha sugerido que el Coxsackievirus B3 (CVB3) es responsable en la mayoría de los casos de la miocarditis viral, que resulta en muerte súbita cardíaca aunque esto aún no está lo suficientemente estudiado [41].

2.4. Deep sequencing - NGS

La secuenciación de nueva generación, (*Next Generation Sequencing - NGS*) es un grupo de tecnologías que permiten secuenciar gran cantidad de segmentos de ADN de manera masiva y en paralelo. Ésta tecnología ha revolucionado la investigación genómica ya que por ejemplo, permite relevar el genoma humano en un día. La tecnología NGS aplicada a microbiología permite caracterizar a los patógenos por su definición genómica más que por los convencionales criterios metabólicos o de morfología. Los genomas de los patógenos definen lo que son, y permiten relevar información sobre la sensibilidad a los medicamentos e informar la relación de diferentes patógenos entre sí. Esto puede usarse para rastrear las fuentes de los brotes de infección [11]. Todas las técnicas NGS permiten secuenciar gran cantidad de fragmentos de ADN de forma paralela en un corto lapso.

Dos conceptos que son importantes para entender el proceso y los resultados obtenidos mediante tecnologías NGS son la cobertura y profundidad. La cobertura (*coverage* o *breadth of coverage*, en inglés) se refiere al porcentaje de bases del genoma de referencia que están siendo secuenciadas una cantidad determinada de veces. La profundidad por otro lado (*depth* o *depth of coverage*) representa el número promedio de veces que cada base en el genoma es secuenciada en los fragmentos de ADN [51].

2.5. Descripción de experimento y conceptos relacionados

En este proyecto se analizan datos de secuencias genómicas obtenidos mediante secuenciación profunda. Por este motivo, los datos de entrada no serán una secuencia única dónde se pueda asignar a cada posición un codón determinado. Por el contrario, lo que se tiene es la frecuencia de cada uno de los sesenta y cuatro codones posibles para cada una de las posiciones de la cadena. Normalizando ésta frecuencia (para de este modo obtener la frecuencia relativa) se consigue en definitiva la probabilidad de aparición de cada codón para cada posición.

En el marco de este proyecto un **experimento** se definirá como una población de virus de una cierta cepa determinada, a la cual, cada cierto número de generaciones se le releva el genoma obteniendo un nuevo pasaje. Dado que el mismo experimento es posible que se realice en más de una ocasión, se agrega para su identificación el número de repetición R. A su vez, las cantidad de posiciones también puede variar. Por lo tanto, el experimento queda caracterizado por la cepa del virus (*strain*), la repetición R, el número de pasajes p (el cual se puede interpretar

Position	Coverage	AAA	AAC	AAG	⋯	⋯	⋯	TTC	TTG	TTT
3	0	0	0	0	⋯	⋯	⋯	0	0	0
6	0	0	0	0	⋯	⋯	⋯	0	0	0
9	0	0	0	0	⋯	⋯	⋯	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
132	66442	0.00010536	0.99856	0	⋯	⋯	⋯	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
7443	0	0	0	0	⋯	⋯	⋯	0	0	0
7446	0	0	0	0	⋯	⋯	⋯	0	0	0
7449	0	0	0	0	⋯	⋯	⋯	0	0	0

Tabla 2.1: Ejemplo de archivo de entrada.

como medida de tiempo transcurrido) y el rango de posiciones válidas. Éste último está determinado por una dupla de valores: FVP y LVP (*First Valid Position* y *Last Valid Position* respectivamente). A modo de ejemplo, en el trabajo anterior se tienen nueve experimentos de tres cepas diferentes (tres repeticiones o réplicas para cada cepa) con veinte pasajes y 2187 posiciones.

En definitiva para cada experimento (repetición R de una cepa determinada), se tendrá p archivos correspondientes a cada uno de los p pasajes. Cada uno de estos archivos es un CSV (Comma Separated Values) donde cada fila corresponde a una posición en la cadena. La primera columna corresponde justamente a la posición, la segunda a la cobertura (o *coverage*), el cual proporciona un estimado de la probabilidad de presencia del nucleótido, y las sesenta y cuatro restantes presentan la frecuencia relativa de cada uno de los codones posibles en esa posición. En la tabla 2.1 se puede ver un ejemplo de un archivo de entrada. Éste mismo formato de archivo es el admitido por el software desarrollado en éste proyecto.

2.5.1. Modelado de la realidad del proyecto

Además de experimento, para el diseño de la herramienta se definen otros conceptos importantes que permiten modelar la realidad. Éstos surgen del modelo conceptual desarrollado utilizando para ello un modelo entidad relación, el cual se puede visualizar en la figura 2.6.

A lo definido anteriormente, para el modelo conceptual, **experimento** queda determinado como una cepa, una repetición, un FVP y un LVP. Éste contiene un conjunto de pasajes, los cuales tienen un número que lo identifica y un conjunto de posiciones. Para cada posición se tiene el conjunto de codones con su frecuencia relativa, una entropía y el número de posición que lo identifica. La entropía, como se explica en el capítulo 3, busca condensar la información de la frecuencia relativa de los codones en un solo valor.

Las **funciones** son las distintas herramientas de análisis y visualización y están definidas por su nombre. Tienen asociados parámetros que utilizan para su ejecución. Éstas trabajan con **instancia experimento** la cual es el **experimento** asociado a un usuario en particular. Por último, la aplicación de las funciones en

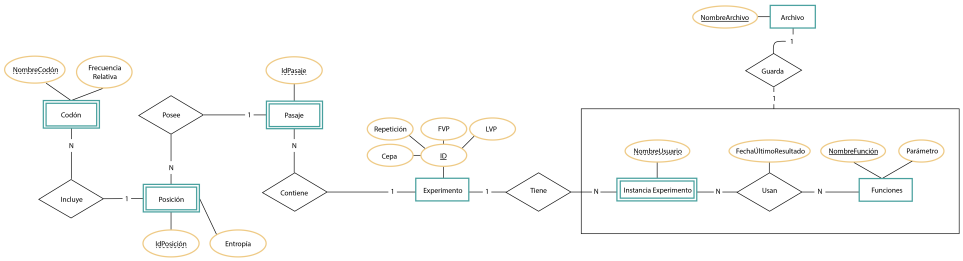


Figura 2.6: Modelo conceptual.

dichas instancias, tienen una fecha de su última aplicación y el resultado se guarda en un **archivo**.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 3

Análisis de los datos y resultados previos

Como ya se ha mencionado, éste proyecto toma como base las herramientas desarrolladas en el proyecto "Análisis y Visualización de la Evolución de Virus" [31]. En dicho proyecto, se analizaron datos de tres cepas de un mismo virus para los cuales se realizaron tres repeticiones (R), relevando su genoma en veinte pasajes (p) cada seis generaciones aproximadamente. De esta manera se completa una totalidad de nueve experimentos.

Se estudiaron tres cepas de virus, donde dos han sido modificadas genéticamente para variar la fidelidad o tendencia a cometer errores durante la replicación viral. La primera es la cepa silvestre que se encuentra en la naturaleza y por tanto sin mutaciones dirigidas a alterar su fidelidad de copia (WT). La siguiente de estas cepas será más proclive a cometer errores y por lo tanto a generar mutaciones, por lo que será denominada de baja fidelidad (Low Fidelity, LF o 299 por la mutación introducida para que ésto ocurra). La tercer cepa es una variante menos proclive a cometer errores y por tanto denominada de alta fidelidad (High Fidelity, HF o 372 por la mutación introducida). Los datos que se utilizaron en dicho proyecto de investigación, fueron curados de una base de datos con secuencias genómicas consistentes en la secuenciación profunda del virus Coxsackie B3. Se secuenciaron genomas completos de pasajes temporales en cultivo celular de éstas tres cepas de poblaciones virales (tres réplicas biológicas por cepa, nueve en total) obteniendo para cada posición del genoma la frecuencia de los codones presentes en la población.

Al comienzo del experimento es cuando las nueve réplicas presentan mayor homogeneidad donde para cada posición, todas las secuencias virales tienen el mismo codón excepto para las posiciones mutadas artificialmente que diferencian las tres cepas. Comenzando de este punto cada réplica evolucionó durante cuarenta pasajes (ciento veinte generaciones aproximadamente) secuenciándose los pasajes impares (1,3,...39) totalizando veinte pasajes. Mediante la secuenciación se obtuvo la frecuencia de aparición de los sesenta y cuatro codones para cada posición (para el caso del virus estudiado, 2187 posiciones).

Durante los pasajes en cultivo celular el virus se debe adaptar a su entorno que

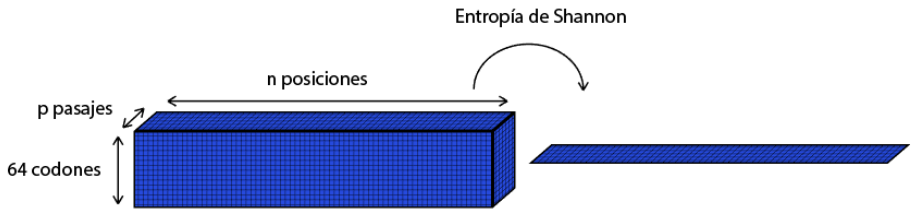


Figura 3.1: Reducción de la dimensión de los datos mediante entropía de Shannon de sesenta y cuatro a uno.

en este caso son células HeLa las cuales provienen de un cáncer cérvico uterino. Dicha adaptación es lograda mediante mutaciones aleatorias de los individuos llegando de esta forma a una población más heterogénea. Aquellas mutaciones que resulten favorables para la adaptación del virus al medio verán incrementada su frecuencia temporalmente.

En éste capítulo se analizarán las distintas aproximaciones metodológicas para analizar datos genómicos de virus desarrollados en el proyecto previo. La implementación de éstos métodos se verá en mayor detalle en el capítulo 6, mientras que las imágenes que acompañan el análisis fueron obtenidas en la implementación de éste proyecto. En primera instancia se estudió la variación de las frecuencias de los codones mayoritarios, el cual es el método tradicional en el área. Como se vió en el capítulo 2 se tienen sesenta y cuatro codones posibles, lo que dificulta la visualización de la evolución de las frecuencias de estos tripletes en dos dimensiones (Frecuencia $\times$ Pasajes (Tiempo)). Por este motivo se consideró la utilización de la Entropía de Shannon para reducir la dimensión de sesenta y cuatro a uno como se observa en la figura 3.1.

En el contexto de la Teoría de la Información, dada una fuente que transmite un conjunto discreto de símbolos, la entropía de Shannon es una medida de la incertidumbre promedio que se tiene sobre el estado de la fuente, basado en las probabilidades de aparición de cada uno de los símbolos de la fuente [13]. En general, dado un conjunto de N posibles eventos de probabilidad de ocurrencia $p_1, p_2, \cdots, p_N$, se tiene que la entropía de Shannon será la dada por la ecuación 3.1,

$$H(z) = \sum_{i=1}^{N} p_i \log_2 p_i \tag{3.1}$$

Viendo la cadena genómica del virus como una fuente de información, tenemos sesenta y cuatro posibles símbolos dados por los codones. Aplicando la fórmula de entropía con $N = 64$, se obtiene la ecuación 3.2

$$H(z) = \sum_{i=1}^{64} p(z_i) \log_2 p(z_i) \tag{3.2}$$

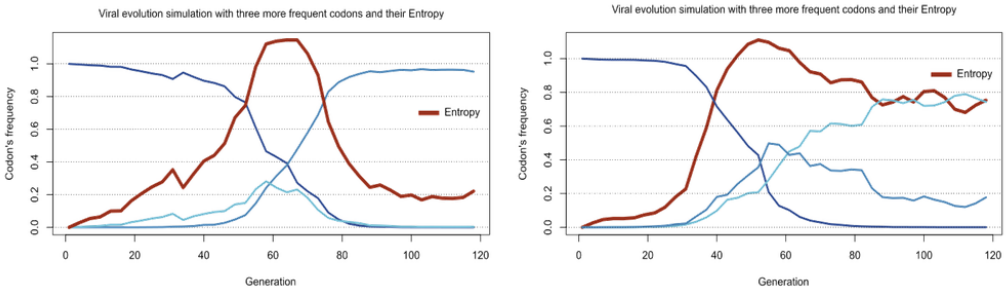


Figura 3.2: Simulación de evolución viral de los tres codones más frecuentes. Gráfica de sus frecuencias y la respectiva entropía. A la izquierda el escenario donde un codón es sustituido por otro. A la derecha el escenario donde un codón es sustituido por dos codones [31].

donde $p(z_i)$ es la frecuencia relativa correspondiente al codón en el índice i.

De esta manera se reduce la información de la frecuencia relativa de los codones en una determinada posición a un sólo valor correspondiente a la entropía de Shannon calculada. Una entropía baja para una posición dada, es indicador de que existe un único codón cuya frecuencia relativa (es decir probabilidad de aparición) es prácticamente 1. Por el contrario, una entropía alta, da cuenta de alta variabilidad. Como se puede ver en la figura 3.2 mediante la entropía se logra captar fenómenos de interés evolutivo relativos a la frecuencia de aparición de los codones. En el caso de la sustitución de un codón por otro, cuando las frecuencias de aparición de los codones es similar (es decir son prácticamente equiprobables) se tendrá un máximo de entropía. Para el escenario donde el codón más frecuente es sustituido por dos codones, se puede observar un crecimiento en la entropía, manteniéndose en un valor alto. En definitiva, analizar la frecuencia de los codones simultáneamente a través de la entropía permite capturar eventos significativos desde el punto de vista evolutivo.

Mapeando la variación de entropía a las regiones de la cápside viral según la posición en el genoma, se puede ver en la figura 3.3 que valores altos de entropía se corresponden a regiones de la parte exterior de la cápside.

A partir de la entropía de Shannon se desarrollaron y evaluaron varios métodos para analizar y visualizar los datos. Se realizó un análisis de la distribución de los máximos de entropía para encontrar posiciones relevantes. Se generaron distintos tipos de visualización, se graficaron las superficies de entropía en tres dimensiones: tiempo (pasajes), posición y entropía; se realizó una descomposición de la entropía para cada posición en variación principal (LV - Leading Variations) y variación aleatoria (RV - Random Variations); y se realizaron los biplots PCA que llevó a hallar posiciones relevantes de cada cepa.

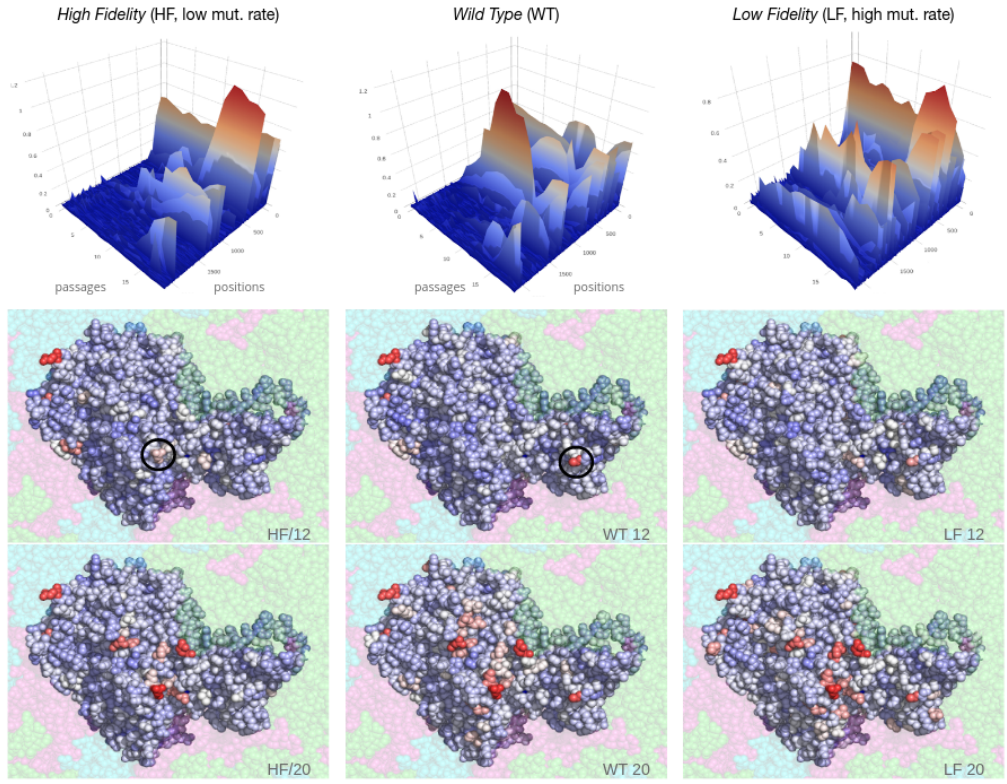


Figura 3.3: Mapeo de los valores de entropía a las regiones de la cápside. De izquierda a derecha, 372 (HF), WT, 299 (LF). Arriba pasaje 12, debajo el pasaje 20. En rojo las posiciones de entropía más alta en el genoma viral [31].

3.1. Análisis mediante variación de frecuencia de codones dominantes

Este método consiste en tomar el primer pasaje del experimento y para cada posición considerar el codón de mayor frecuencia. Luego se sigue la variación en frecuencia de dicho codón en esa posición. Es decir se construye un vector de tamaño n, cantidad de pasajes (en el caso de los datos analizados, $n = 20$) siguiendo la frecuencia de ese codón en esa posición. Se establece un umbral (entre 0 y 1 ya que se está considerando una frecuencia relativa o probabilidad). Aquellas posiciones para las cuales la frecuencia relativa de su codón dominante sea menor que dicho valor serán destacadas como posibles posiciones de interés.

En la figura 3.4, utilizando los datos del experimento correspondiente a la cepa 299, repetición 3, con posiciones comprendidas entre $[774, 7332]$ y umbral 0,7 se puede observar que se destacan las posiciones $[1995, 2469, 3054, 3444, 5613]$.

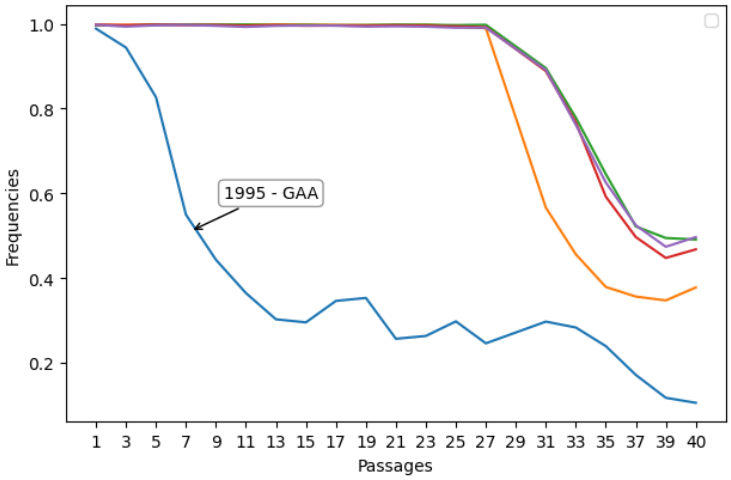


Figura 3.4: Variación de frecuencia de codón dominante. Análisis de posiciones de interés mediante variación de frecuencia de codones dominantes. Umbral = 0,7. Cepa 299 repetición 3.

3.2. Definición de un espacio de variaciones

Considerando la trayectoria de la evolución de la entropía a través de los pasajes se tiene una serie temporal o una señal. De éste modo se tiene que para cada posición k se tendrá una serie temporal de la forma, $X_k = [x_1, x_2, \cdots, x_n]$ con k variando entre las posiciones posibles (para los datos analizados se tienen 2187 posiciones variando entre $[774, 7332]$) y n la cantidad de pasajes (en el caso de los datos estudiados, $n = 20$). Observando ésta señal, como se puede ver en la figura 3.5, se pueden distinguir dos componentes que afectan la evolución viral. La primera es debida a la tasa mutacional del virus y es la causante de que se introduzcan sustituciones de manera aleatoria. La segunda componente es introducida por la fijación de aquellas mutaciones que implican una mejor adaptación al medio.

Bajo el supuesto de que estas dos componentes son procesos con diferente tasa temporal (y por tanto, ancho de banda diferentes) se buscó descomponer la trayectoria mediante un filtrado pasabajos. De esta forma se obtienen dos señales, una variación suave que se denominó Leading Variations (LV) y un residuo al que se denominó Random Variations (RV). Para ello se utilizó un promediado con ventana deslizante de tamaño $m = 7$ descomponiendo la trayectoria original en $X_k^{LV} = [x_1^{LV}, x_2^{LV}, \cdots, x_n^{LV}]$ y $X_k^{RV} = [x_1^{RV}, x_2^{RV}, \cdots, x_n^{RV}]$. Cada uno de los valores x_i^{LV} está dado por la ecuación 3.3

$$x_i^{LV} = \frac{1}{m} \sum_{j=i-m/2}^{i+m/2} x_j \tag{3.3}$$

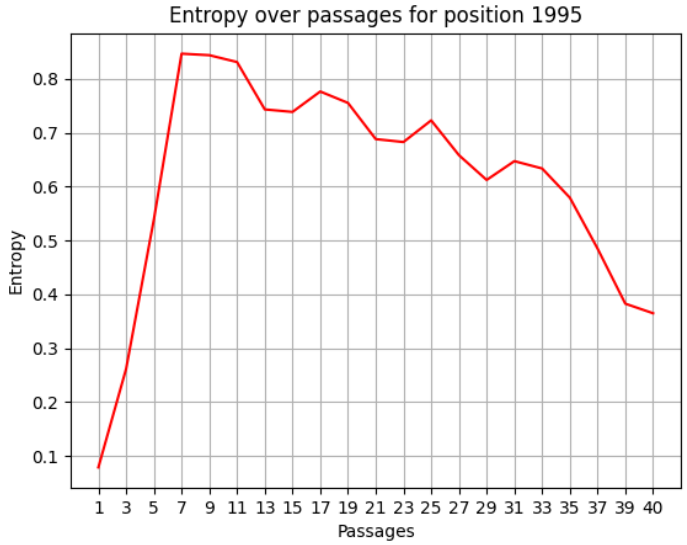


Figura 3.5: Evolución de la entropía en el tiempo(pasajes) para la posición 1995 de la cepa 299, repetición 1.

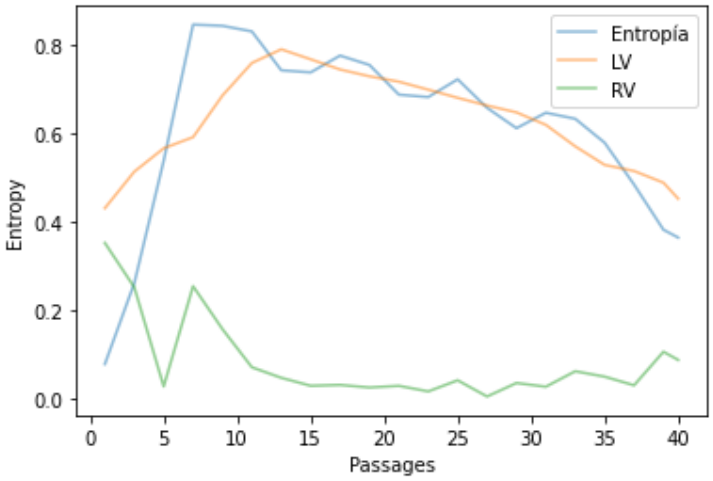


Figura 3.6: Descomposición LV-RV. Componentes LV y RV para el experimento de la cepa 299 repetición 1, posición 1995.

Mientras que los valores x_i^{RV} se calculan como indica la ecuación 3.4

$$x_i^{RV} = x_i^{LV} - x_i \tag{3.4}$$

Así, como se puede observar en la figura 3.6 las LV son las que contienen la mayor información de la señal original y presentan mayor similitud con ella, mientras que las RV muestran un comportamiento más aleatorio y con poca semejanza a la trayectoria original.

A partir de estas dos componentes obtenidas mediante el filtrado pasabajos por ventana deslizante, para cada posición se calcula el acumulado en cada una de

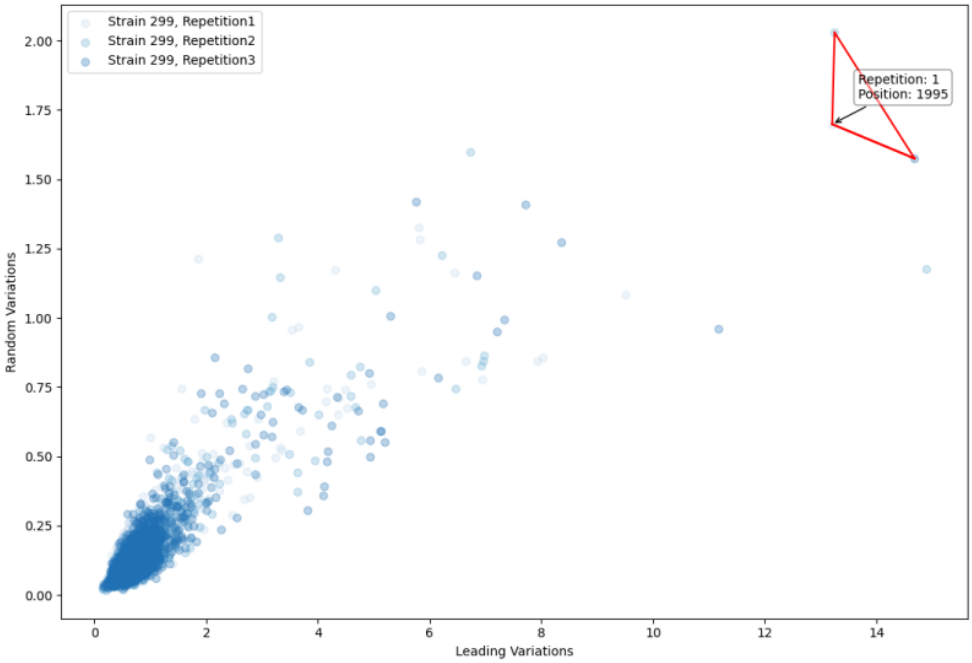


Figura 3.7: Análisis de espacio de variaciones para la cepa 299 con sus tres repeticiones.

esas curvas y se representa cada una de ellas como un punto en el espacio LV-RV como se muestra en las ecuaciones 3.5 y 3.6

$$LV = \sum_{j=1}^{n} x_j^{LV} \tag{3.5}$$

$$RV = \sum_{j=1}^{n} x_j^{RV} \tag{3.6}$$

con n cantidad de pasajes.

En ésta representación como se puede observar en la figura 3.7, la mayoría de las posiciones se encuentran en valores bajos de acumulados tanto de LV como de RV.

3.3. Análisis estadístico de los máximos de la entropía

Analizando las superficies de entropía de la figura 3.8, se observa que existen posiciones donde se percibe un alto cambio en su valor. Este cambio en la entropía está asociado a un cambio en la distribución de la frecuencia de los codones. Por ejemplo en una posición donde había un codón dominante pasan a observarse varios codones con una frecuencia relativamente alta. Con el fin de encontrar éstos

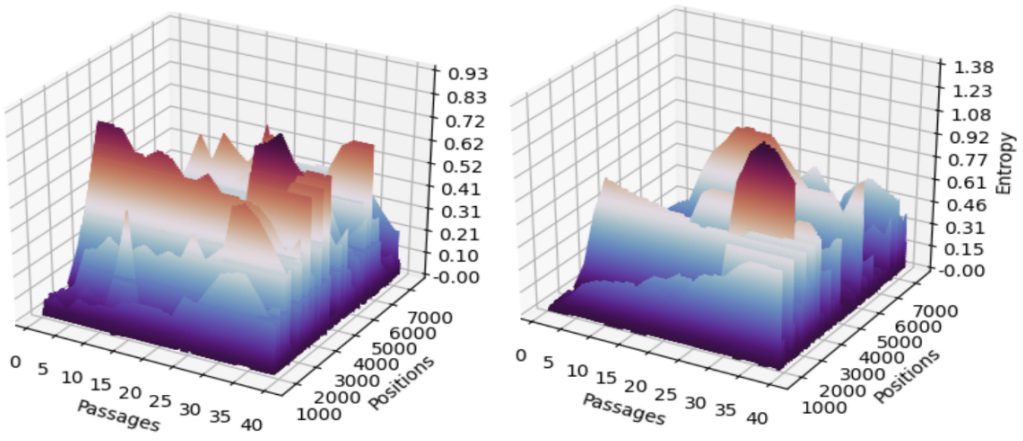


Figura 3.8: Superficies de Entropía para la cepa 299 y repetición 1 y cepa WT y repetición 3, ambas con posiciones en el intervalo $[774, 7332]$.

eventos en los datos se realizó un análisis de la distribución de los máximos de entropías.

Para ello se ajusta la distribución empírica de los máximos de entropía a una distribución teórica utilizada para modelar máximos. Sean $\{X_1, X_2, ...X_p\}$ el máximo de entropía en los n pasajes para las p posiciones del genoma viral, se asume que las X_i son variables aleatorias independientes e idénticamente distribuidas con distribución Gumbel de parámetros μ y β. En el caso de los datos analizados, se tiene $n = 20$ y $p = 2187$. Dicha distribución es una de las típicamente utilizadas para modelar eventos de máximos y tiene la ecuación 3.7

$$F(x; \mu, \beta) = e^{(-e^{-(x-\mu)/\beta})} \tag{3.7}$$

Se calcula el cuantil que separa la distribución en 95 % y 5 % de los datos. La cola por derecha a dicho cuantil (es decir, los datos más extremos) se descartan. Realizar dicho corte tiene sentido ya que se espera que las posiciones donde actúa la selección natural tengan un comportamiento diferente al resto de los máximos. A partir de estos datos se genera un histograma obteniéndose una distribución. Se ajustan los parámetros de la distribución de Gumbel a los datos mediante el método de máxima verosimilitud y se grafica la distribución de Gumbel junto al histograma como se puede observar en la figura 3.9.

Se asume dicha distribución para lo máximos de entropía y se calculan los *p-valores* para cada una de las posiciones de la cadena. Las diez posiciones cuyo *p-valor* resulta más pequeño son señaladas como posibles posiciones de interés.

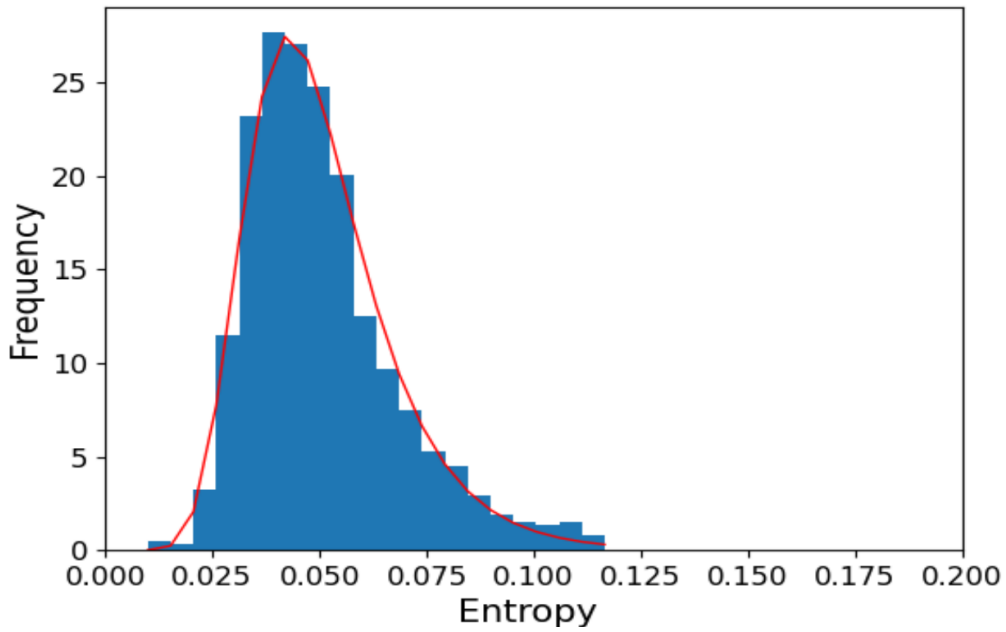


Figura 3.9: Análisis de estadístico de los máximos de entropía para el experimento de cepa 299, repetición 2.

3.4. Detección de codones representativos de las cepas mediante representación de la evolución viral en baja dimensión

Con el fin de analizar los datos en baja dimensión se realizó un Análisis de Componentes Principales (PCA por sus siglas en inglés). Se tomaron las nueve réplicas considerando cada una de ellas como un punto de dimensión 43740 (2187 posiciones $\times$ 20 pasajes). En el caso general se tendrían puntos de dimensión $p \times n$ con p cantidad de posiciones y n cantidad de pasajes. Se graficaron lo nueve puntos utilizando las dos primeras dimensiones (los dos primeros vectores propios) del análisis de componentes principales.

Se graficaron las coordenadas de las 43740 variables en los dos vectores propios. Lo que se busca es conocer qué variables influyen con el fin de identificar las cepas. Cada una de las 43740 variables del vector de cada réplica corresponde a una dupla $[Posición, Pasaje]$

A partir de esta gráfica se puede ver en la figura 3.10 que los puntos que están acumulados en dirección hacia cada una de las cepas muestran curvas de trayectoria de entropía que permiten distinguir dicha cepa. De esta forma se pueden obtener posiciones donde las cepas se comportan de manera diferente, lo cual se puede interpretar como posible indicador de que esté actuando la selección. De la misma manera, los puntos acumulados en el centro presentan trayectorias de entropías similares para las tres cepas como se puede observar en la figura 3.10.

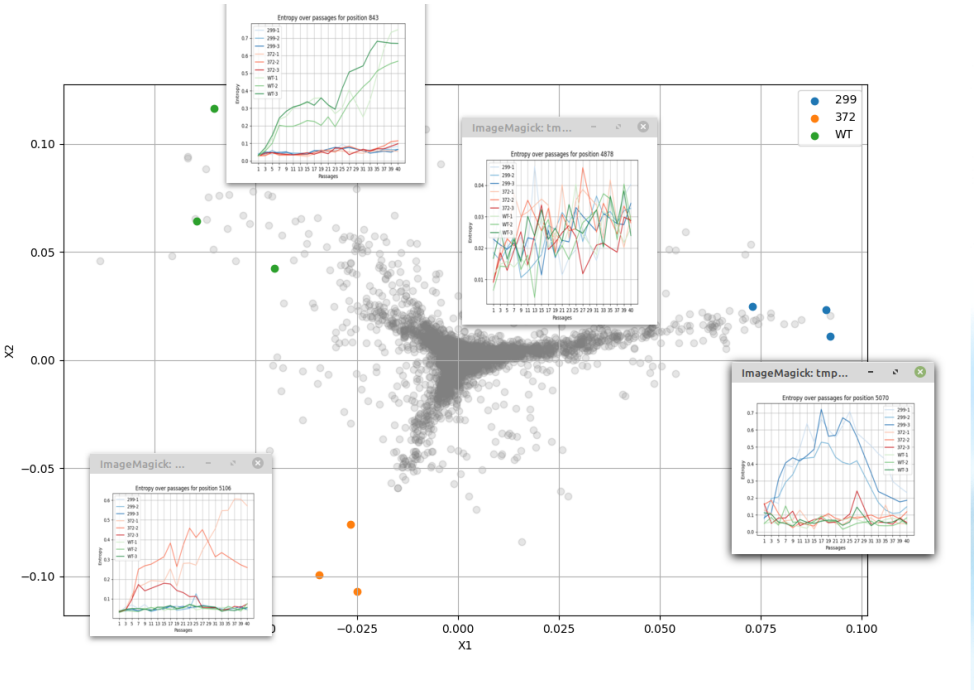


Figura 3.10: Descomposición PCA para las tres cepas con sus respectivos tres experimentos. Se grafica la trayectoria de la entropía para las posiciones cercanas a la representación de los experimentos (los nueve puntos de colores) mostrando como cercano a estos puntos, se encuentran posiciones con entropías altas correspondientes a estas cepas y en el centro, que son similares las entropías.

Capítulo 4

Visualización mediante t-SNE

Uno de los objetivos de este proyecto es brindarle al usuario herramientas de visualización que le puedan dar información relevante. Con este fin se desarrolló el módulo de t-SNE, donde se aplica éste algoritmo a los datos evolutivos del virus y de este modo se logran distintas visualizaciones. Para ello se utilizó la biblioteca existente de Python scikit-learn. En las secciones siguientes se explicará en qué consiste el método t-SNE para luego mostrar cómo se aplicó en el problema tratado.

4.1. Algoritmo t-SNE

t-SNE (t-distributed Stochastic Neighbor Embedding) es un algoritmo de reducción de dimensiones no lineal basado en SNE, utilizado principalmente para visualizar datos de alta dimensión. El algoritmo apunta a que tanto las distancias locales como globales en alta dimensión se vean reflejadas en el mapeo en baja dimensión. Como se puede observar en la figura 4.1, el ejemplo muestra datos en dimensión dos que son mapeados a dimensión uno. Lo que busca el algoritmo es reducir la dimensión de los datos (en este caso a dimensión uno), pero al mismo tiempo mantener tanto las distancias locales (agrupar los puntos de mismo color) como globales (mantener las distancias entre las distintas agrupaciones).

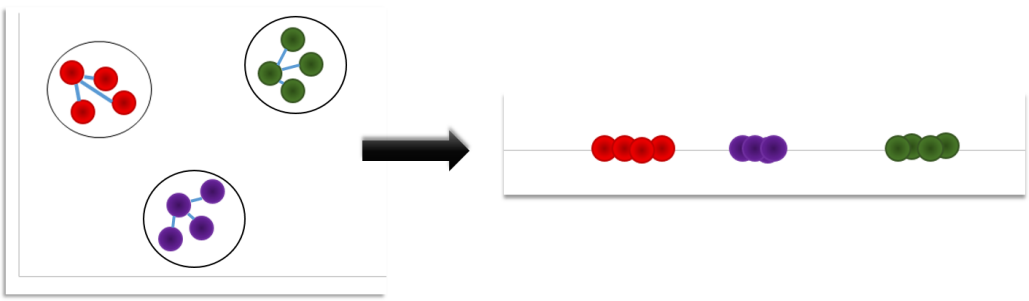


Figura 4.1: Conjunto de puntos en dos dimensiones mapeados a una dimensión [55].

En este contexto, se denomina $X = \{x_1, x_2, ..., x_n\}$ al conjunto de puntos en alta dimensión los cuales se desean mapear a los puntos $Y = \{y_1, y_2, ..., y_n\}$ en baja dimensión. Dado que el algoritmo es utilizado principalmente para visualización de datos, por lo general se reduce la dimensión de los mismos a dos o tres dimensiones. El algoritmo lo que plantea es, primero calcular una medida de similitud entre puntos en alta dimensión (cuán probable es que sean vecinos, o cuán similar es x_i a x_j). Luego se crea una conjunto de puntos en baja dimensión y nuevamente, se calcula una medida de similitud entre ellos (cuan probable es que sean vecinos, o cuan similar es y_i a y_j). Por último, se calcula cuan ajustado es el mapeo, y a través de un proceso iterativo, se modifican los puntos en baja dimensión $Y = \{y_1, y_2, ..., y_n\}$ hasta que las similitudes entre los puntos de este conjunto reflejen las similitudes entre los puntos en alta dimensión $X = \{x_1, x_2, ..., x_n\}$.

Para obtener esta medida de similitud entre puntos en alta dimensión, t-SNE transforma la distancia Euclídea en probabilidades condicionales. Esta medida cuantifica cuan similares son dos puntos o, lo que es lo mismo, que probabilidad hay de que sean vecinos. Para ello se centra una Gaussiana en x_i y así, la probabilidad condicional $p_{j|i}$ de que x_i sea vecino de x_j será dada por dicha distribución. De esta manera, se tendrá probabilidad $p_{j|i}$ relativamente alta para puntos cercanos y por el contrario, para puntos alejados $p_{j|i}$ será muy pequeña.

De este modo, se tiene que la probabilidad condicional $p_{j|i}$ está dada por la ecuación 4.1

$$p_{j|i} = \frac{exp(-\|x_i - x_j\|^2/2\sigma_i^2)}{\sum_{k \neq i} exp(-\|x_i - x_k\|^2/2\sigma_i^2)} \tag{4.1}$$

siendo σ_i la varianza de la Gaussiana centrada en el punto x_i. Además, se toma $p_{i|i}$ como cero dado que interesa solamente modelar similitudes de a pares. En la figura 4.2 se puede ver una representación gráfica de éste cálculo.

Para determinar la varianza σ_i de la Gaussiana centrada en cada punto de alta dimensión x_i, se debe tener en cuenta que la densidad de los puntos en el espacio de alta dimensión puede variar, y por tanto, difícilmente se encuentre un único valor de σ_i que se ajuste a todos los puntos. Valores pequeños de σ_i serán ajustados para zonas de alta densidad, pero no para zonas de baja densidad. Dado el valor de σ_i seleccionado queda determinada la distribución de probabilidad P_i, sobre todos los demás puntos. t-SNE realiza una búsqueda binaria para encontrar el valor de σ_i que produce un P_i con un *perplexity* fijo especificado por el usuario donde dicho parámetro está definido por la ecuación 4.2

$$Perp(P_i) = 2^{H(P_i)} \tag{4.2}$$

siendo $H(P_i)$ la entropía de Shannon de P_i en bits,

$$H(P_i) = -\sum_j p_{j|i} log_2(p_{j|i}) \tag{4.3}$$

Dado que esta distribución tiene una entropía que incrementa a medida que σ_i incrementa, el parámetro *perplexity* puede ser interpretado como una medida del número de vecinos considerados [30].

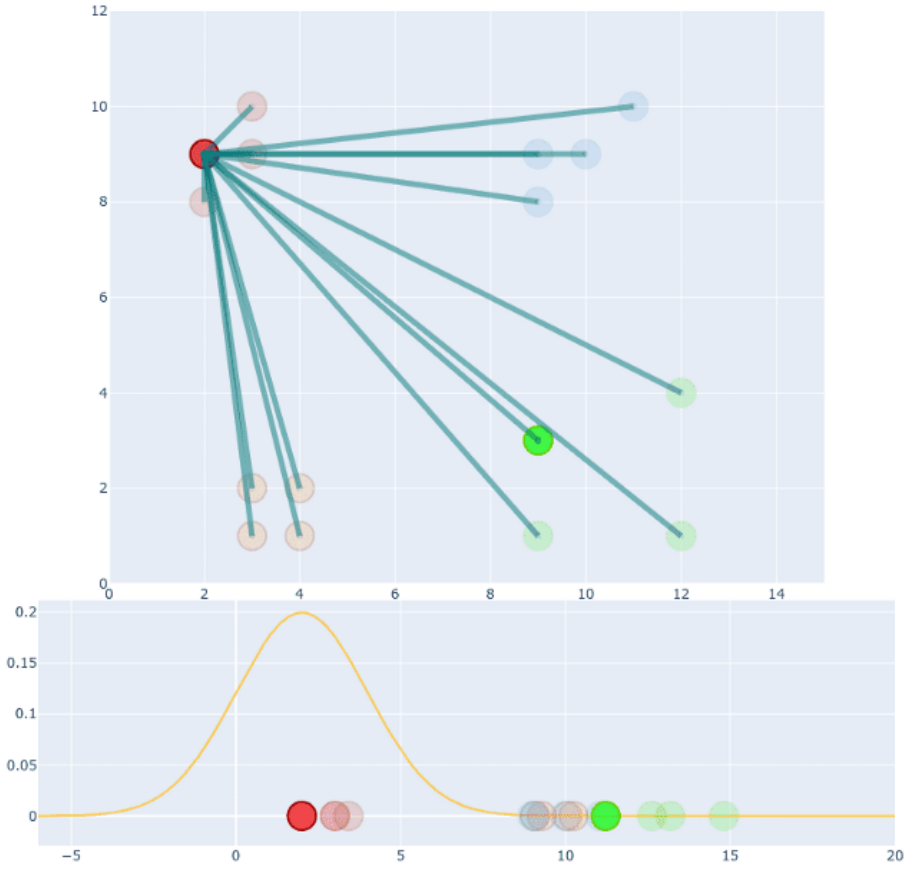


Figura 4.2: Gaussiana centrada en el punto de interés para cálculo de probabilidad condicional [25].

A partir de las probabilidades condicionales anteriores, se calcula la probabilidad conjunta 4.4

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n} \tag{4.4}$$

Esta probabilidad conjunta es la que se utilizará como medida de similitud entre puntos en el espacio de alta dimensión. Además tiene la propiedad de que $p_{ij} = p_{ji}$. Una de las diferencias con SNE es la utilización de esta probabilidad conjunta en vez de la probabilidad condicional.

Se procede a crear los puntos en el espacio de baja dimensión de forma aleatoria y de manera similar a lo realizado en el espacio de alta dimensión, se calcula la similitud entre puntos. En este caso, en vez de utilizar una distribución Gaussiana para la probablilidad conjunta se utiliza una distribución t de Student (de ahí la

t en el nombre del algoritmo). De este modo, la probabilidad conjunta está dada por la ecuación 4.5

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l}(1 + \|y_k - y_l\|^2)^{-1}} \tag{4.5}$$

Teniendo una medida de similitud entre puntos tanto en alta como en baja dimensión. se estima cuan bueno es el mapeo realizado. Para ello se utiliza la divergencia de Kullback-Leibler la cual permite cuantificar que tan similares son dos distribuciones de probabilidad. Cuanto más pequeña sea la divergencia de Kullback-Leibler, mayor similitud entre las distribuciones de probabilidad. Así, lo que se busca es minimizar esta cantidad para lograr que el mapeo en baja dimensión refleje las distancias o similitudes en alta dimensión. Así, la función de costo que se debe minimizar está dada por la ecuación 4.6

$$C = KL(P||Q) = \sum_i \sum_j p_{ij} log\left(\frac{p_{ij}}{q_{ij}}\right) \tag{4.6}$$

De este modo, el gradiente tiene la forma de la ecuación 4.7

$$\frac{\partial C}{\partial y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + \|y_k - y_l\|^2)^{-1} \tag{4.7}$$

Mediante el método del descenso del gradiente se van modificando de manera iterativa los puntos en baja dimensión para minimizar la función de costo. Así, los cambios en las coordenadas de los puntos mapeados son modificadas de acuerdo a la ecuación 4.8. Para ello, al gradiente actual, se agrega un término de suma de gradientes previos que decae exponencialmente (mediante un término multiplicativo) para así evitar caer en mínimos locales. Matemáticamente se tiene que,

$$\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\partial C}{\partial \mathcal{Y}} + \alpha(t)(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}) \tag{4.8}$$

con $\mathcal{Y}^{(t)}$ la solución en la iteración t, η es el *learning rate* y $\alpha(t)$ el término agregado en la iteración t para que la suma de gradientes previas caiga exponencialmente.

Así, t-SNE modela puntos poco similares en distancias grandes en baja dimensión, y puntos similares en distancias pequeñas en baja dimensión.

4.2. t-SNE con Scikit-learn

Como se mencionó, t-SNE convierte similitudes entre los puntos en probabilidades. Las similitudes en el espacio de alta dimensión son representadas por probabilidades conjuntas Gaussianas, mientras que las del espacio de baja dimensión se representan por probabilidades conjuntas con distribución t de Student. De esta forma t-SNE permite extraer la estructura local de los datos y a su vez resaltar agrupaciones de puntos.

Para obtener una mejor visualización de los datos, la implementación de scikit-learn del algoritmo t-SNE permite manejar varios parámetros que se describen a continuación.

Maximum number of iterations controla como su nombre lo indica la cantidad de iteraciones máxima permitida. Por defecto se toma como 1000, siendo éste valor suficientemente grande como para que generalmente no sea necesaria su modificación.

Early exageration factor corresponde a un término multiplicativo de las probabilidades p_{ij} en las primeras etapas de la optimización. Como se mencionó anteriormente, éstas probabilidades corresponden a las del espacio de alta dimensión. Esto causará que prácticamente todas las q_{ij} sean muy pequeñas para modelar a las p_{ij} teniendo como efecto que las agrupaciones presentes en los datos tiendan a formar grupos compactos y muy separados. Este parámetro generalmente tampoco es necesario realizar su ajuste.

Perplexity modifica la cantidad de vecinos considerados por t-SNE para generar las probabilidades. Valores de *perplexity* altos llevan a considerar mayor cantidad de vecinos, y por tanto menor sensibilidad a la estructura de los datos local. Por el contrario, valores pequeños de *perplexity* considera menor cantidad de vecinos y por tanto prioriza la estructura local de los datos por sobre la global. Cabe destacar que para datasets más grandes, se requerirán mayor cantidad de vecinos para relevar la estructura local, y por tanto valores de *perplexity* más grandes serán necesarios.

Learning Rate éste parámetro afecta el gradiente del algoritmo. Si es muy pequeño puede provocar que el algoritmo pare en un mínimo local no óptimo. Si es muy grande incrementará la divergencia de Kullback-Leibler durante la optimización.

En definitiva, a partir de la implementación de scikit-learn del algoritmo t-SNE, se utilizaron dos parámetros para ajustar y lograr una buena visualización de los datos, *Perplexity* y *Learning Rate* [40].

4.3. Aplicación de t-SNE a los datos

Considerando cómo funciona t-SNE y su objetivo, se aplicó éste algoritmo de visualización a los datos genómicos de virus con la intención de brindarle al usuario información respecto a la estructura de los mismos.

Con este fin, se trabajó con los datos ordenándolos de dos modos diferentes para luego lograr la visualización con el algoritmo.

Primero se tomó cada pasaje como un punto. De esta manera para cada experimento se tiene tantas filas como pasajes. A su vez, estas filas tiene tantas columnas como posiciones. En conclusión cada punto que se toma para el algoritmo t-SNE tiene dimensión la cantidad de posiciones del experimento En el caso de los datos disponibles se tienen 20 puntos de dimensión 2187.

En la figura 4.3 se puede ver resaltado en rojo la manera en que se ordenaron los datos para su visualización t-SNE. Cada fila es un punto del conjunto de datos.

	Posición 1	Posición 2	Posición 3	...	...	...	Posición N
Pasaje 1	E_{11}	E_{12}	E_{13}	...	...	...	E_{1N}
Pasaje 2	E_{21}	E_{22}	E_{23}	...	...	...	E_{2N}
Pasaje 3	E_{31}	E_{33}	E_{33}	...	...	...	E_{3N}
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Pasaje n	E_{n1}	E_{n2}	E_{n3}	...	...	...	E_{nN}

Figura 4.3: Puntos de entrada para t-SNE por pasajes.

Ésto se realiza con los parámetros Perplexity y Learning Rate variando en ciertos valores fijados por la herramienta (3-6-12-15 para el Perplexity y 10-100-300-500 para el Learning Rate). Estos valores se fijaron en base a las visualizaciones de los datos obtenidas y teniendo en cuenta los valores típcos para estos dos parámetros. A su vez se le da la posibilidad al usuario de ingresar un nuevo par de valores permitiendo obtener nuevas visualizaciones. Lo que se busca en este caso es permitirle al usuario encontrar similitudes entre los pasajes de los diferentes experimentos seleccionados para así encontrar estados de la población viral que se asemejen.

Como se puede ver en la figura 4.4, utilizando los valores de 6 para el *Perplexity* y 100 para el *Learning Rate*, la visualización t-SNE realizada por pasajes agrupa de acuerdo a las cepas. Las dos agrupaciones que se resaltan en rojo corresponden a los primeros pasajes de cada cepa (el más grande corresponde a los pasajes 1 y 3, mientras que el pequeño corresponde al pasaje 5). Para el caso de los pasajes 1 y 3, los nueve puntos de dichos pasajes están cercanos. En el caso del pasaje 5, son siete de los nueve puntos los que se encuentran cercanos. Esto tiene sentido ya que en los primeros pasajes es cuando más similitudes tienen las nueve poblaciones de virus.

La segunda visualización que se realizó utilizando t-SNE, es tomando cada trayectoria de entropía por posición como un punto. Es decir se tendrán tantos puntos como cantidad de posiciones, y cada uno de esos puntos tendrá como dimensión la cantidad de pasajes. Para el caso de los datos analizados se tienen 2187 puntos de dimensión 20.

En la figura 4.5 se puede ver resaltado en rojo la manera en que se ordenaron los datos en este caso para pasarlos al algoritmo de visualización t-SNE. Cada fila es un punto del conjunto de datos.

Nuevamente la idea es reflejar las similitudes de los datos en alta dimensión (en este caso la alta dimensión es 20) en el mapeo de baja dimensión (dos dimensiones en este caso). Con éste método se busca encontrar posiciones cuya trayectoria de entropía sea lo suficientemente diferente al resto como para que se pueda visualizar en el espacio de baja dimensión.

Considerando que las posiciones relevantes desde el punto de vista evolutivo para estos datos han sido identificadas previamente en el proyecto que antecede [31], se resaltan en el plano t-SNE dichas posiciones para así visualizar cómo quedan distribuidas por el algoritmo. Las posiciones son $[1392, 1995, 2379, 4830, 4863, 5070]$. En la figura 4.6 obtenida utilizando valores de *Perplexity*=12 y *Learning Rate*=300, se las puede ver resaltadas en rojo. Se observa que quedan relativamente agrupadas

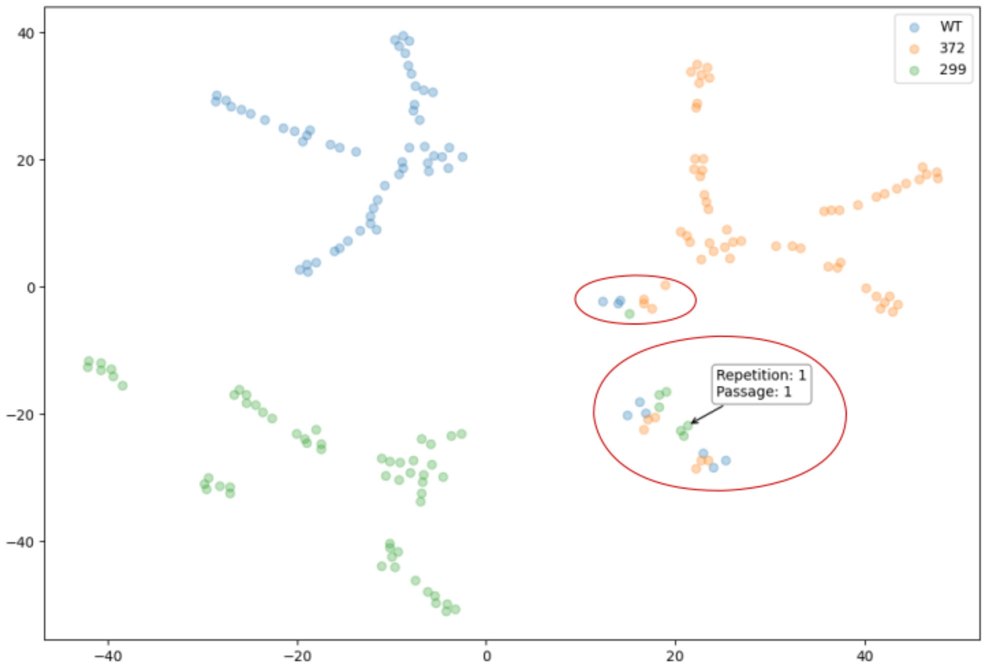


Figura 4.4: Visualización t-SNE por pasajes con valores *Perplexity*=6 y *Learning Rate*=100. Se observa que los datos son agrupados de acuerdo a su cepa y que en los primeros pasajes, es cuando los puntos de cada cepa más se asemejan.

	Pasaje 1	Pasaje 2	Pasaje 3	...	...	...	Pasaje n
Posición 1	E_{11}	E_{21}	E_{31}	...	...	...	E_{n1}
Posición 2	E_{12}	E_{22}	E_{23}	...	...	...	E_{n2}
Posición 3	E_{13}	E_{33}	E_{33}	...	...	...	E_{n3}
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Posición N	E_{1N}	E_{2N}	E_{3N}	...	...	...	E_{nN}

Figura 4.5: Puntos de entrada para t-SNE por posiciones.

y no centralizadas dentro del mapeo t-SNE. Como se muestra en la figura para la posición 1995, la herramienta permite obtener la información de cada punto y de esta manera lograr que a partir de la estructura en t-SNE el usuario encuentre aquellos puntos no centralizados o distantes como posibles puntos de interés.

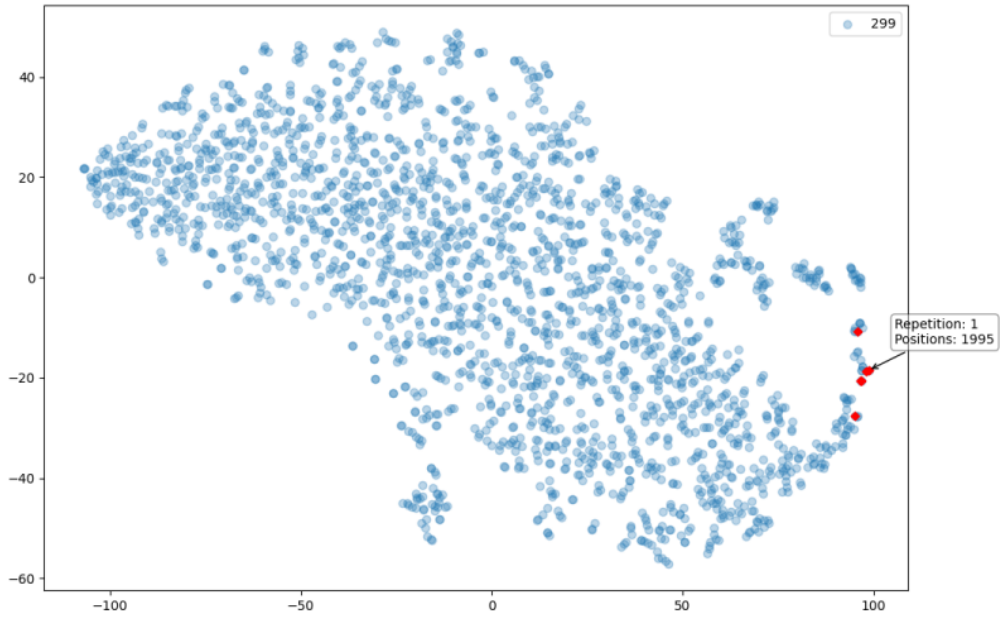


Figura 4.6: t-SNE por posiciones aplicado a la cepa 299 repetición 1. Se resaltan en rojo las posiciones de interés relevadas en el proyecto previo.

Capítulo 5

Agrupación y descomposición de series temporales

Además de la visualización t-SNE, se brinda al usuario una nueva herramienta de análisis de los datos para así obtener y analizar información de forma de encontrar las posiciones que pueden resultar de interés desde el punto de vista adaptativo del virus.

Para cada experimento, se tiene por posición una trayectoria de la evolución de la entropía y por tanto estos datos son una serie temporal. Se busca realizar agrupaciones de trayectorias similares y de esta manera encontrar posiciones de interés biológico. Por lo tanto, para cada experimento se tendrán tantas trayectorias como posiciones en el genoma viral, donde cada una de estas trayectorias tiene tantos puntos como pasajes el experimento.

Se analizaron dos aproximaciones para agrupar las trayectorias de entropías. Si bien se utilizó el mismo algoritmo, los datos de entrada para cada uno de ellos son diferentes. Al primero se le pasa la trayectoria de entropía por posición tal cual fue computada. En el segundo caso se realiza una descomposición de componentes de la serie temporal y se le pasa la primera de ellas al método de agrupamiento. En definitiva se puede ver como un filtrado pasabajos previo a la entrada al método de agrupamiento. En las secciones siguientes se explica el método de agrupamiento utilizado, la descomposición que se emplea y los posteriores resultados obtenidos al aplicarlos a los datos.

5.1. Agrupación de series temporales

Como se mencionó, la evolución de la entropía a través de los pasajes es una serie temporal. Ésto quiere decir que cada uno de los puntos tiene un orden cronológico el cual debe ser tenido en cuenta a la hora de realizar un análisis y extraer información de ella. Por esta razón se eligió un método de agrupamiento que tenga en cuenta ésta propiedad y de esta forma poder realizar un análisis cabal de las diferentes trayectorias.

El algoritmo elegido para realizar la agrupación es k-means. Dado un grupo de muestras de tamaño n, éste método consiste en particionar la totalidad de muestras n en k agrupaciones. Se asigna a qué grupo pertenece cada muestra de acuerdo a la distancia hacia el centroide o media de cada uno de los agrupamientos. La distancia hacia el centroide que resulte más pequeña será el grupo asignado al dato [26].

La elección de la métrica a utilizar es de gran relevancia en éste caso. Dado que se desea tener en cuenta que se trata de una serie temporal, utilizar la distancia euclidiana no parece ser la mejor opción. Teniendo esto en cuenta se utiliza el algoritmo de agrupamiento k-means usando la métrica DTW (Dynamic Time Warping) la cual se explica en la siguiente sección.

5.2. Dynamic Time Warping

DTW es una medida de similitud entre series temporales que busca desestimar diferencias en tiempos entre ellas para devolver valores pequeños entre curvas cuya trayectoria es similar (a pesar de no estar alineadas en el tiempo) [57]. Dadas dos series temporales $x = (x_0, ..., x_{n-1})$ e $y = (y_0, ...y_{m-1})$ de largos n y m respectivamente, con x_i e y_i de dimensión d, el valor de DTW está dado por la ecuación 5.1

$$DTW(x, y) = \min_{\pi} \sqrt{\sum_{(i,j)\in\pi} d(x_i, y_j)^2} \tag{5.1}$$

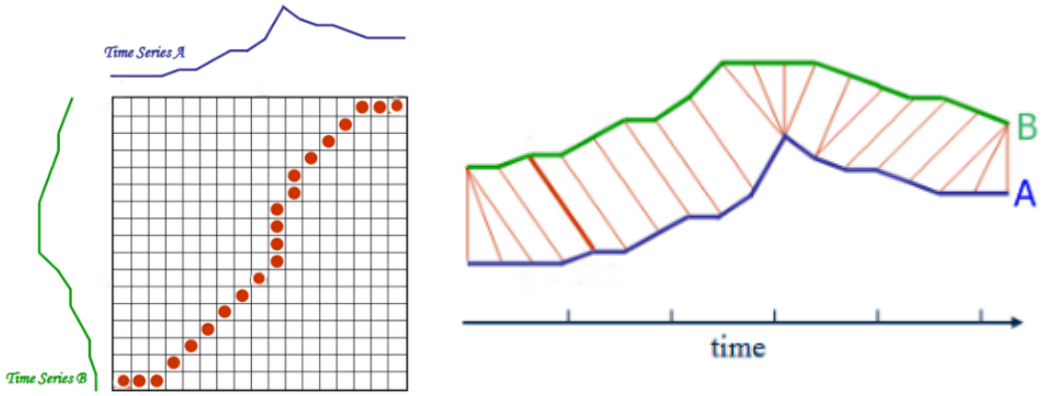
donde π es la alineación de las series que permite obtener la distancia Euclideana mínima. Esta alineación π es denominada camino y está constituida por $\pi = [\pi_0, ..., \pi_K]$. Cada uno de los $\pi_k = (i_k, j_k)$ es un par de índices que cumple $0 \leq i_k < n$, y $0 \leq j_k < m$. Es decir, i_k representa un elemento de la serie temporal x e y_k un elemento de la serie temporal y. Además $\pi_0 = (0, 0)$ y $\pi_k = (n-1, m-1)$, esto significa que el primer índice de la primer secuencia debe estar alineado con el primer índice de la segunda, y el último índice de la primer secuencia debe estar alineado con el último índice de la segunda secuencia. Por último, $\forall k > 0$, se tiene que $\pi_k = (i_k, j_k)$ y $\pi_{k-1} = (i_{k-1}, j_{k-1})$ cumplen con lo siguiente,

- $i_{k-1} \leq i_k \leq i_{k-1} + 1$
- $j_{k-1} \leq j_k \leq j_{k-1} + 1$

En definitiva, lo que se hace es recorrer todos lo elementos de ambas series temporales (alineando el primer y último índice de ambas) de manera continua y monótonamente creciente.

Dado que DTW no cumple con la desigualdad triangular, no clasifica como distancia. Sin embargo sí cumple las siguientes propiedades,

- $\forall x, y\ DTW(x, y) \geq 0$
- $\forall x\ DTW(x, x) = 0$



(a) Camino DTW (b) Correspondencia del camino en las series temporales

Figura 5.1: En la figura a se puede ver el camino DTW y en la figura b, la correspondencia en la serie temporal [54].

En la figura 5.1 se pueden ver dos series temporales donde se resalta el camino encontrado por DTW y la correspondencia entre los distintos puntos de las dos series temporales.

Si se utiliza la métrica DTW, el cálculo de los centroides de los distintos grupos se corresponde con encontrar el μ que cumpla lo siguiente,

$$\min_{\mu} \sum_{x \in \mathcal{D}} DTW(\mu, x)^2 \tag{5.2}$$

De esta manera, el algoritmo k-means trabaja iterativamente asignando a qué grupos pertenece cada trayectoria de acuerdo a la distancia a los centroides para luego encontrar los nuevos centroides de cada grupo [57].

5.3. Descomposición de la serie temporal - Singular Spectrum Analysis

Para analizar los datos mediante agrupaciones de las trayectorias, a través de otro enfoque, se buscó descomponer la serie temporal en distintas componentes. Se busca estudiar los efectos de dicha descomposición en la agrupación obtenida.

La descomposición elegida fue Signal Spectrum Analysis [20] la cual consiste en una extracción mediante componentes principales. De esta manera se puede descomponer la serie temporal $X = x_1, x_2, ..., x_N$ en componentes independientes buscando encontrar estructura de tendencia, componentes cíclicos, estacionales y ruido. Éstos componentes son obtenidos a través de una descomposición propia de la matriz de trayectoria de la serie temporal.

Dada una serie temporal de valores reales de tamaño N, $X = x_1, x_2, ..., x_N$ con N > 2. Siendo X una serie no nula (es decir, existe al menos un x_i que cumple que $x_i \neq 0$), el primer paso de la descomposición consiste en tomar subsecuencias de dicha serie temporal. Para ello, se toma L con $1 < L < N$ y $K = N - L - 1$, denominado largo de ventana. De esta forma se obtienen vectores de largo L, los cuales son subsecuencias X_i de las serie temporal X dados por la ecuación 5.3

$$X_i = (x_i, \cdots, x_{i+L-1})^T \quad (1 \leq i \leq K) \tag{5.3}$$

A partir de estos vectores se forma la matriz de trayectoria

$$X = [X_1 : \cdots : X_K] = \begin{pmatrix} x_1 & x_2 & x_3 & \cdots & x_K \\ x_2 & x_3 & x_4 & \cdots & x_{K+1} \\ x_3 & x_4 & x_5 & \cdots & x_{K+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_L & x_{L+1} & x_{L+2} & \cdots & x_N \end{pmatrix} \tag{5.4}$$

Ésta matriz tiene dimensión $L \times K$ ya que sus columnas son los vectores subsecuencias X_i. Se puede ver que tanto las columnas como las filas de dicha matriz son subseries de la original.

El segundo paso consiste en realizar la descomposición Singuar Value Descomposition (SVD) de la matriz de trayectoria formada anteriormente. Sean $S = XX^T$ con valores propios $\lambda_1, \cdots, \lambda_L$ ordenados de manera decreciente ($\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_L \geq 0$) y $U_1, \cdots U_L$ el sistema ortonormal de autovectores de la matriz S correspondientes a dichos autovectores. Siendo $d = rango(X) = max\{i \text{ tal que } \lambda_i > 0\}$ (usualmente se tiene que $d = min\{L, K\}$) y $V_i = X^T U_i / \sqrt{\lambda_i} \quad (i = 1, \cdots, d)$ [20].

Así, la descomposición SVD de la matriz de trayectoria X está dada por 5.5

$$X = X_1 + \cdots + X_d \tag{5.5}$$

con $X_i = \sqrt{\lambda_i} U_i V_i^T$ matrices de rango 1. A estas matrices se las denomina matrices elementales. El conjunto $(\sqrt{\lambda_i}, U_i, V_i)$ es denominado *eigentriple*.

El tercer paso es agrupar las matrices de la descomposición SVD. Se particiona los índices $1, \cdots, d$ en m subconjuntos disjuntos $I_1, \cdots, I_m$. Sea $I = i_1, \cdots, i_p$ la matriz resultante X_I correspondiente a el conjunto I es definida como $X_I = X_{i_1} + \cdots + X_{i_p}$. Las matrices resultantes son calculadas para los conjuntos $I_1, \cdots, I_m$ por lo que se tiene que,

$$X = X_{I_1} + \cdots + X_{I_m} \tag{5.6}$$

Si $m = d$ se tienen j matrices I_j con $j = 1, \cdots, d$. A este agrupamiento se lo denomina elemental.

El cuarto paso consiste en transformar cada matriz X_{I_j} de la ecuación 5.6 en una serie temporal de tamaño N mediante el promediado de la diagonal. Sea Y una matriz $L \times K$ con elementos y_{ij}. Sea $L^* = min\{L, K\}$, $K^* = max\{L, K\}$, $N = L + K - 1$ y sea $y^*_{ij} = y_{ij}$ si $L < K$, y $y^*_{ij} = y_{ji}$ de lo contrario. Para transformar la matriz Y en una serie de largo N $y_1, \cdots y_N$ se utiliza la siguiente

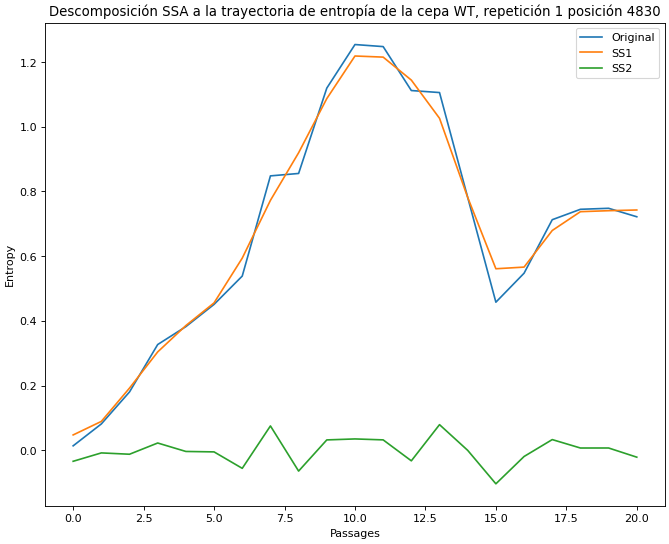


Figura 5.2: SSA aplicado a una de las trayectorias de entropía.

ecuación,

$$y_k = \begin{cases} \frac{1}{k}\sum_{m=1}^{k} y^*_{m,k-m+1} & para \quad 1 \leq k \leq L^* \\ \frac{1}{L^*}\sum_{m=1}^{L^*} y^*_{m,k-m+1} & para \quad L^* \leq k \leq K^* \\ \frac{1}{N-k+1}\sum_{m=k-K^*+1}^{N-K^*+1} y^*_{m,k-m+1} & para \quad K^* \leq k \leq N \end{cases} \tag{5.7}$$

Esta fórmula corresponde a promediar los elementos de la matriz sobre sus antidiagonales: $i+j = k+1$. Aplicando la ecuación de promediado de las diagonales 5.7 a las matrices resultantes X_{I_k} se llega a una serie reconstruida $\tilde{\mathbb{X}}^{(k)} = (\tilde{x}_1^{(k)}, \cdots, \tilde{x}_N^{(k)})$. De este modo la serie inicial $x_1, \cdots x_N$ es descompuesta en una suma de m series reconstruidas,

$$x_n = \sum_{k=1}^{m} \tilde{x}_n^{(k)} \quad (n = 1, 2, \cdots, N) \tag{5.8}$$

En la figura 5.2 se puede ver la aplicación de la descomposición SSA a una de las trayectorias de entropía (posición 4830 de la repetición 1 de la cepa WT).

5.4. Aplicación del enfoque de series temporales a los datos

Como se mencionó en las secciones previas se estudiaron dos agrupaciones. Por un lado se utilizaron las trayectorias de entropía para encontrar las agrupaciones, y por otro se realizó una descomposición de dichas trayectorias pasando al algoritmo la primer componente de dicha descomposición. En las secciones siguientes se analizan los resultados obtenidos a partir de dichos estudios.

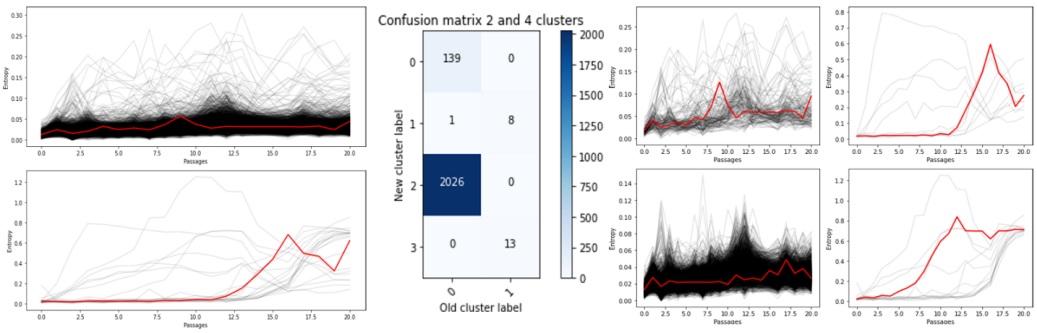


Figura 5.3: Agrupación en dos y cuatro grupos de la repetición 1 de la cepa WT.

5.4.1. Agrupaciones con las trayectorias de entropía

Para realizar el estudio de agrupaciones de las trayectorias de entropía de cada experimento se seleccionó el algoritmo k-means con métrica DTW. Se realizó la agrupación para dos, cuatro, seis, ocho y diez grupos. A partir de estos resultados, se grafican las trayectorias de entropía para cada uno de estos grupos con su respectivo centroide. Además se estudió de qué manera se van formando los grupos a partir de una matriz de confusión.

Se utilizó el experimento WT repetición 1 para graficar los resultados obtenidos de las diferentes agrupaciones. Como se puede ver en la figura 5.3 para el caso de agrupar en dos grupos se tiene uno mayoritario y otro donde aparecen solamente 21 trayectorias de las 2187 posibles. Se puede distinguir que el grupo minoritario tiene aquellas trayectorias con valores más grandes de entropía.

De las siete posiciones marcadas como relevantes en el trabajo previo cinco se encuentran dentro de éstas 21$[1392, 1995, 2379, 4830, 4863]$.

La matriz de confusión permite visualizar como se van descomponiendo las agrupaciones a medida que crece la cantidad de grupos. En el eje horizontal se puede ver la etiqueta asignada previamente, y en el eje vertical la nueva etiqueta asignada cuando el número de grupos aumentó. Así, por ejemplo en la figura 5.3 se puede ver que 13 curvas de trayectorias que tenían etiqueta 1 en la agrupación de dos grupos pasan a tener etiqueta 3 en la agrupación con cuatro grupos. De este modo, a partir de la matriz de confusión para dos y cuatro grupos de la figura 5.3 se puede ver que el grupo pequeño original prácticamente se parte en dos. Lo mismo ocurre con el grande pero en menor medida. El resultado son dos grupos muy pequeños donde se acumulan las trayectorias bien diferentes, mientras en los otros dos se siguen viendo trayectorias de entropía pequeña. De hecho, por las trayectorias de la figura 5.3 se observa que el agrupamiento sigue separando de acuerdo al nivel de entropía de las trayectorias.

Para el caso de seis grupos, como se visualiza en la figura 5.4 nuevamente se separan de los dos más poblados algunas trayectorias que pueden resultar de interés. En esta agrupación se puede ver un nuevo punto de partida a partir del que se irá desmembrando a grupos más pequeños. Los dos más pequeños permanecen prácticamente iguales.

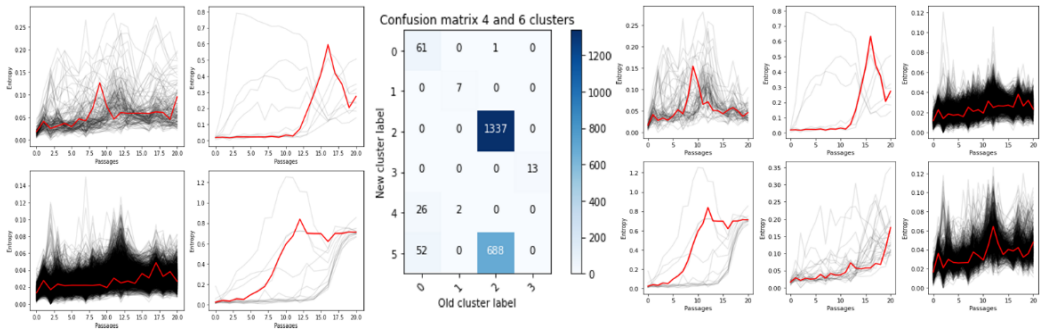


Figura 5.4: Agrupación en cuatro y seis grupos de la repetición 1 de la cepa WT.

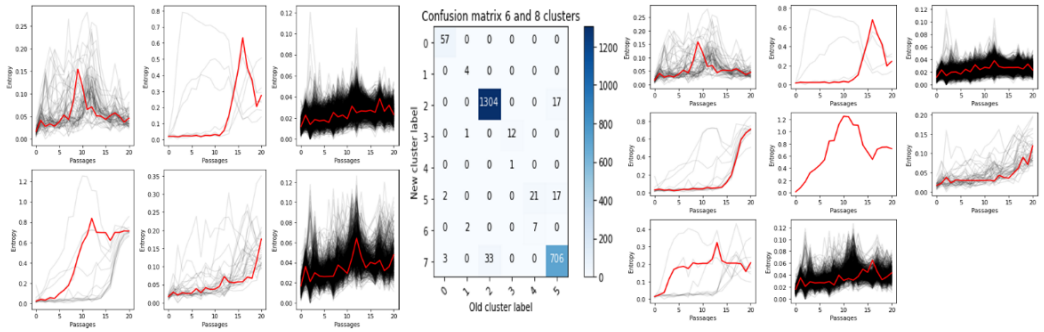


Figura 5.5: Agrupación en seis y ocho grupos de la repetición 1 de la cepa WT.

Se puede ver a partir de las matrices de confusión y de las trayectorias que hasta seis agrupaciones, el algoritmo separa trayectorias que pueden interesar. Para mayor cantidad, lo que sucede es que se separan en subgrupos aún más pequeños los grupos menos poblados. Esto se puede observar a partir de las figuras 5.5 y 5.6

Por este motivo se decidió incluir en la herramienta la agrupación para dos, cuatro y seis agrupaciones. A su vez aquellos grupos con cardinal menor o igual al 1 % del total de posiciones del genoma son considerados como de interés, y por tanto sus posiciones desplegadas en pantalla.

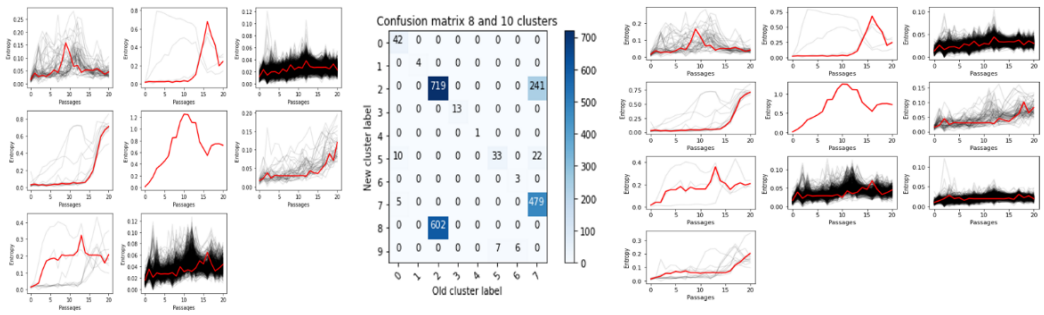


Figura 5.6: Agrupación en ocho y diez grupos de la repetición 1 de la cepa WT.

Aplicando éste algoritmo a los datos, en los nueve experimentos se devuelven 112 posiciones de interés. De ellas, solamente cuatro fueron señaladas en los nueve experimentos $[1392, 1995, 2469, 4830]$. La posición 4863 fue apuntada en ocho de los nueve experimentos.

Discriminando por cepa, además de las mencionadas anteriormente, se tiene que en el caso de la 299, las posiciones 4863 y 5070 son apuntadas en los tres experimentos. Para la cepa 372, la posición 2379 es señalada en las tres repeticiones. Finalmente para las cepa silvestre (WT), las posiciones $[843, 1956, 2379, 2733, 4863]$ son discriminadas como posiciones atípicas en los tres experimentos.

5.4.2. Agrupación con Singular Spectrum Analysis

Para el caso de la agrupaciones utilizando la descomposición SSA, el comportamiento es similar al anterior. Nuevamente se utilizó la repetición 1 de la cepa WT. Las componentes SSA se calculan utilizando ventana de largo 2 y realizando la agrupación elemental. Se utilizó el algoritmo k-means con distancia DTW.

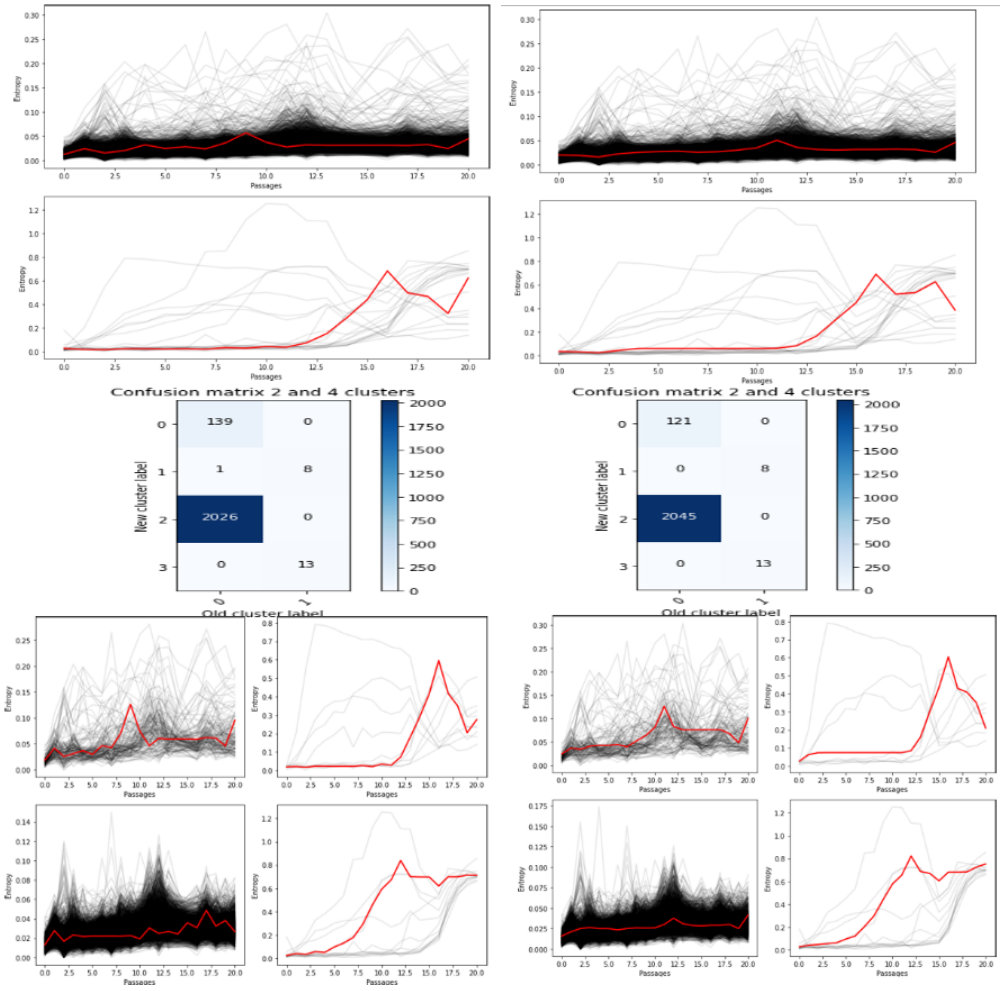
Observando las figuras 5.10b y 5.10 podemos concluir que la descomposición captura la estructura de la señal ya que hasta la agrupación de seis grupos tanto las trayectorias como la respectiva matriz de confusión son muy similares.

Si bien, para el caso de ocho y diez agrupaciones se observan diferencias entre las agrupaciones obtenidas, aquellas que contienen pocas posiciones y con valores de entropía alta son muy similares. En el caso de SSA se logra agrupar un conjunto de trayectorias cuya entropía es baja (no supera el valor de 0,3) pero que tienen una forma con un máximo de aproximadamente dicho valor. Si bien se podría pensar que una trayectoria de esta forma puede llegar a representar una sustitución de un codón por otro, difícilmente sea el caso. La transición de un codón dominante a otro, no se dará de un pasaje a otro, sino que se debería ver las frecuencias de uno aumentando y disminuyendo la del otro, llevando a que la entropía esté en un valor más alto que 0,3.

Discriminando por cepa, además de las mencionadas anteriormente, se tiene que en el caso de la 299, las posiciones 4863 y 5070 son apuntadas en los tres experimentos. Para la cepa 372, la posición 2379 es señalada en las tres repeticiones. Finalmente para las cepa silvestre (WT), las posiciones $[843, 1956, 2379, 2733, 4863]$ son discriminadas como valores atípicos en los tres experimentos.

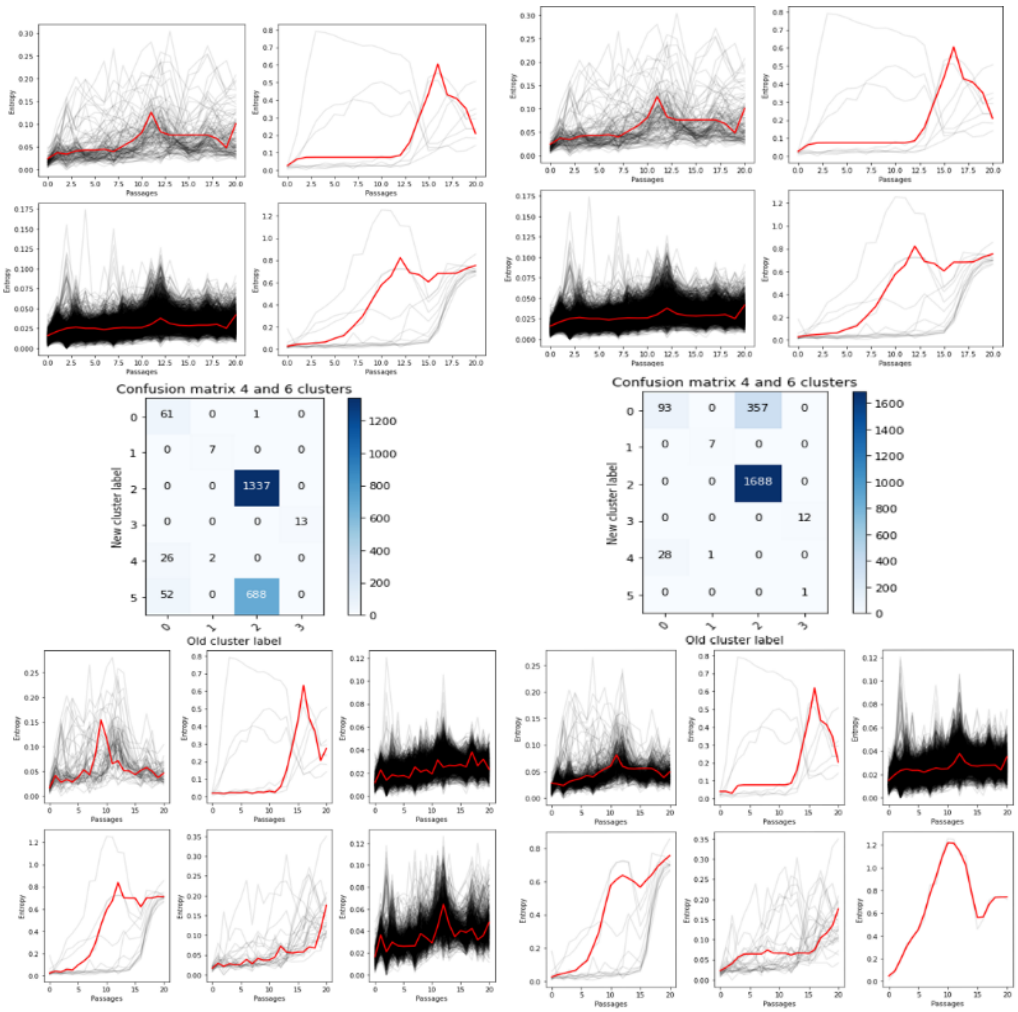
Con esta agrupación, en el caso de los datos utilizados, se comprobó que se seleccionan 123 como posiciones de posible interés en los nueve experimentos. De estas 123, sólo tres fueron seleccionadas en los nueve experimentos: $[1392, 1995, 4830]$. Además, las posición 2469 fue apuntada en ocho de los nueve experimentos, mientras que la posición 4863 en siete de los nueve.

Analizando por cepa, se tiene que para la cepa 299, las posiciones $[2469, 4863, 5070]$ aparecen en los tres experimentos. Para la cepa 372, la posición 2379 se señala en los tres experimentos. Por último para la cepa silvestre (WT), las posiciones $[843, 1956, 2379, 2469, 2733, 4863, 7056]$ se marcan en los tres experimentos.



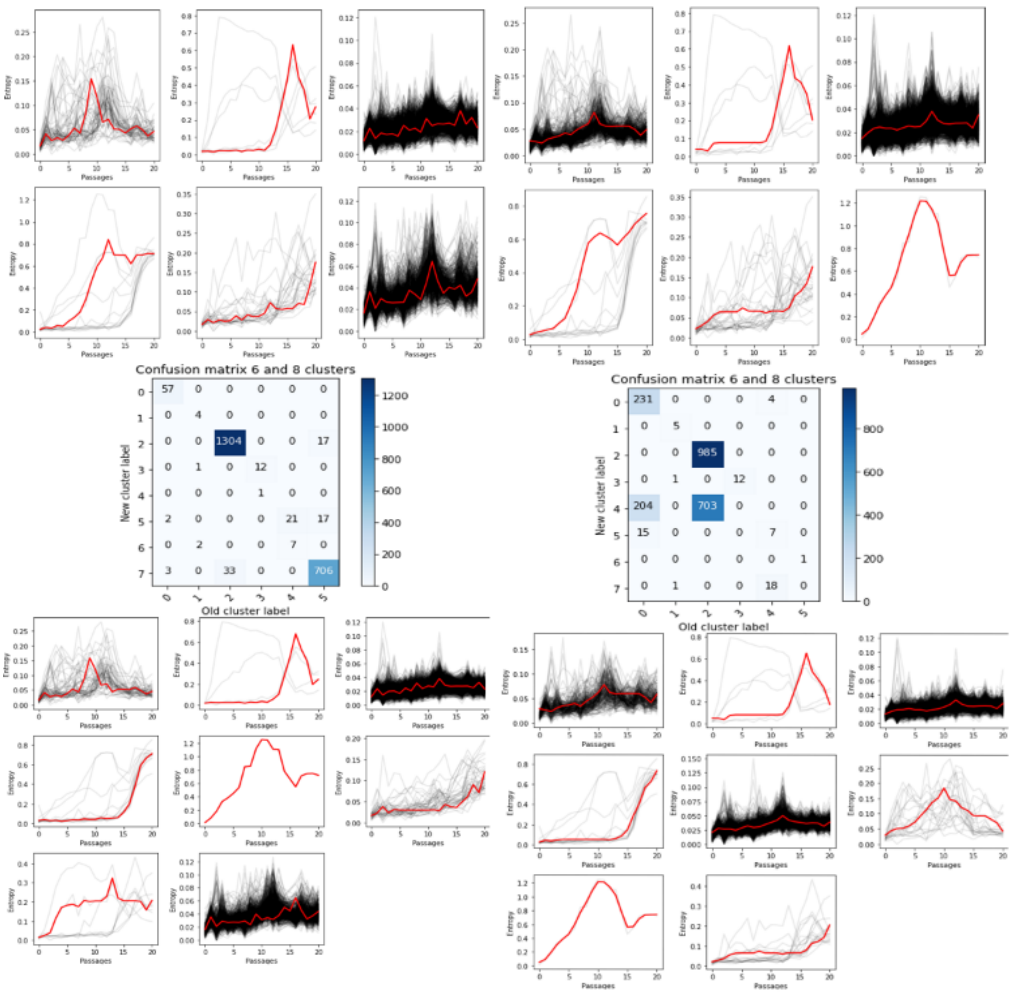
(a) Entropías (b) SSA entropías

Figura 5.7: Comparativa de agrupaciones para dos y cuatro grupos utilizando las trayectorias reales y la descomposición SSA.



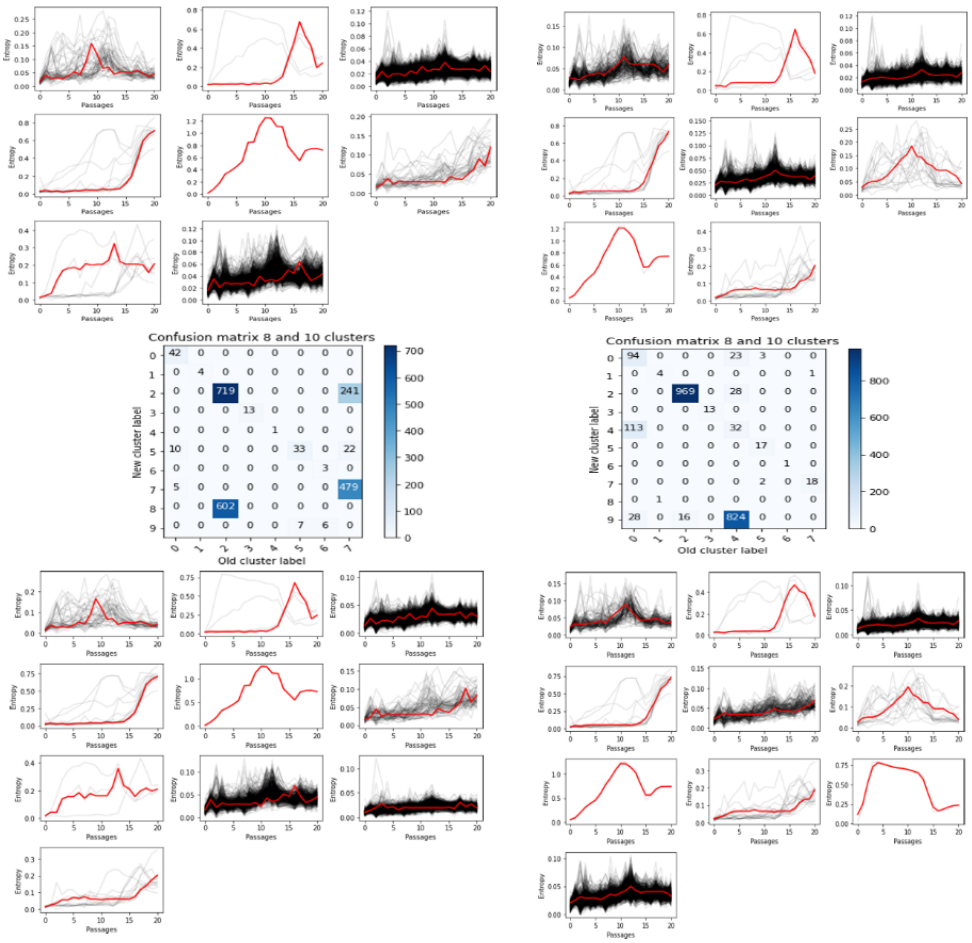
(a) Entropías (b) SSA entropías

Figura 5.8: Comparativa de agrupaciones para cuatro y seis grupos utilizando las trayectorias reales y la descomposición SSA.



(a) Entropías (b) SSA entropías

Figura 5.9: Comparativa de agrupaciones para seis y ocho grupos utilizando las trayectorias reales y la descomposición SSA.



(a) Entropías (b) SSA entropías

Figura 5.10: Comparativa de agrupaciones para ocho y diez grupos utilizando las trayectorias reales y la descomposición SSA.

Capítulo 6

Diseño y desarrollo de la herramienta

En este capítulo se trata las fortalezas y debilidades de la solución existente, los requerimientos de la nueva herramienta, así como las decisiones de diseño e implementación y los métodos implementados. Por último se detallan las distintas maneras de distribuir la herramienta.

6.1. Fortalezas y debilidades de la solución existente

Como se menciona anteriormente en el proyecto que antecede se desarrollaron los métodos analizados en el capítulo 3. Éstos están implementados en código Python y R, y resuelven las necesidades específicas del problema. La solución existente posee ciertas fortalezas y debilidades, las cuales se detallan a continuación.

6.1.1. Fortalezas

En cuanto a fortalezas, en primer lugar está **adaptada a los datos**. Esto quiere decir, que dado el conocimiento respecto a los mismos, se realizan cálculos específicos previos a la ejecución de los métodos de forma de corregir la información. A modo de ejemplo, se realiza la lectura de los datos entre posiciones específicas predefinidas y se interpolan pasajes.

Referente al anterior punto, al estar adaptada a los datos, posee **una única estructura** para almacenar toda la información en disco. Se generan arreglos para almacenar toda la información, y la misma luego de procesada, se almacena en un único archivo.

Está implementado de forma **modular**. Existe un módulo *test* el cual invoca los distintos métodos de forma secuencial.

6.1.2. Debilidades

Si bien se mencionó como fortaleza que la solución está **adaptada a los datos**, esto llevó a una debilidad en la representación de los datos elegida. Al conocer la información del conjunto de datos, se mantienen estructuras así como también

variables preconfiguradas para el análisis de éstos. Esto imposibilita su adaptación a otro conjunto de datos. A modo de ejemplo, se mantienen arreglos creados con una estructura predefinida.

De acuerdo al ejemplo del punto anterior, se utilizan arreglos de Numpy con cuatro índices para poder acceder a la información. Esta estructura **dificulta mantener el código simple** y de esta forma, mantenible. Como ejemplo, el conjunto de datos del proyecto anterior contiene tres cepas con tres repeticiones cada una. En la estructura se tiene un índice de 0 a 2 para acceder a la información de la cepa, así como también para la repetición, lo que propicia errores así como también dificulta la comprensión del código para alguien ajeno al proyecto.

Otra debilidad es que **no posee una interfaz gráfica** donde el usuario pueda interactuar con los distintos métodos. Éstos deben ser invocados a demanda por el usuario, o desde el módulo *test* el cual invoca todos los métodos. Además, los métodos desarrollados generan **visualizaciones estáticas** lo que imposibilita al usuario de obtener más información que la desplegada. Por ejemplo, no es posible obtener información de un punto en una gráfica en particular.

Por último, esta herramienta carece de **registros**. Éste punto es importante, debido a que el usuario podría perder u olvidar información de análisis y de esta forma no poder replicar resultados obtenidos.

6.2. Requerimientos de la herramienta a desarrollar

Este proyecto tiene como grandes requerimientos funcionales, proporcionar una interfaz de usuario, implementar los métodos del proyecto anterior y que éstos puedan trabajar con datos de virus ARN genéricos y proporcionar nuevas herramientas de análisis y visualización de evolución de virus. Éstos se descomponen en otros más específicos.

Para los métodos del proyecto anterior se tiene como requerimientos: **graficar trayectoria de entropía** de una posición dada por el usuario, graficar **superficie de entropía** y **graficar distribución de máximos de entropía**, listando la información de posiciones más atípicas. Para todos éstos se debe almacenar el resultado. También se debe graficar la descomposición **PCA**. Además, se debe graficar las trayectorias de entropía de los experimentos utilizados para una posición elegida por el usuario de ésta descomposición. En el caso de **Codon Decay** se debe graficar la evolución de la dupla $(codón, posición)$ del codón dominante si su frecuencia relativa es inferior al umbral seleccionado por el usuario en algún instante. Otro requerimiento es permitir al usuario interpolar los datos de un experimento para un pasaje elegido. Esta interpolación puede ser por frecuencias o entropía. Por último, se debe graficar las posiciones en el eje de coordenadas LV-RV. El usuario podrá elegir el tipo de aproximación, es decir, promedio móvil o por polinomio y seleccionar la ventana o el grado respectivamente. Además, se deberá mostrar la información de los puntos y si hubiese puntos con igual $(cepa, posición)$, destacarlos.

A partir de las nuevas técnicas desarrolladas en este proyecto, se generaron nue-

vos requerimientos. Para **t-SNE** se debe permitir al usuario seleccionar el tipo, es decir, aplicado a pasajes o posiciones. Además, el usuario debe poder interactuar con los parámetros *Learning Rate* y *Perplexity*. También debe tener la posibilidad de ocultar información de otros experimentos y obtener información de los puntos graficados. Por último, debe poder exportar e importar la información. Con la técnica de **agrupamiento de series temporales** se debe permitir al usuario seleccionar el tipo de series temporales, así como también interactuar con el número de agrupaciones y el tipo de visualización. Se deberá listar información de las posiciones en los grupos mas pequeños así como también poder exportar la información de este etiquetado.

Para todos los métodos anteriormente descriptos, el usuario podrá trabajar con cualquiera de los experimentos cargados en la sesión. Podrá cargar información por directorio de forma de cargar un experimento no procesado y elegir las posiciones válidas del mismo. También tendrá la posibilidad de cargar experimentos ya procesados. Se deberá proveer una interfaz de usuario donde el mismo pueda trabajar en forma paralela con los distintos métodos.

Por último, la herramienta debe poder registrar la actividad de usuario, así como también permitir al usuario obtener información de los análisis efectuados por él.

A nivel de requerimientos no funcionales, la herramienta debe funcionar mínimamente en Linux y además deberá ser software libre. Por último, la misma deberá ser mantenible, es decir, fácilmente modificable para corregir errores, mejorar rendimiento u otros aspectos. En concordancia con esto último, el código deberá ser escrito en ingles, así como también sus comentarios.

6.3. Arquitectura general de la solución

La arquitectura de la solución consta de una aplicación que se ejecuta locamente en el equipo del usuario. Ésta ejecuta el procesamiento de los datos (los cuales se encuentran localmente), así como también es la encargada de la interacción con el usuario. Además, se cuenta con una base de datos no relacional ubicada en la nube.

Esta aplicación consta de tres capas lógicas, las cuales se detallan a continuación [29]:

- **Capa de presentación** la cual presenta el sistema al usuario, le presenta la información y captura las entradas. Esta capa se comunica únicamente con la capa de negocio.

- **Capa de negocio** la cual se comunica con la capa de presentación de forma de recibir las solicitudes de usuario y presentar los resultados. Se encarga de validar la información ingresada y procesar los distintos métodos. También se comunica con la capa de persistencia, para solicitar la información o almacenarla.

- **Capa de persistencia** es donde residen los datos y es la encargada de acceder a los mismos. Recibe solicitudes de la capa de negocio para almacenar o recuperar datos.

Estas capas presentan independencia lógica entre ellas, es decir, si por ejemplo hubiese un cambio de estructura de almacenamiento de los datos, la lógica de la capa de negocios no es afectada [38]. No existe independencia física entre las capas, ya que como se menciona anteriormente, la capa de presentación, la capa de negocio y parte de la capa de persistencia se encuentra localmente en la aplicación.

En las próximas secciones se detallará los componentes así como las distintas decisiones tomadas en cada una de estas capas lógicas.

6.3.1. Capa de presentación

En esta sección se detallan las decisiones tomadas referentes a esta capa, desde qué bibliotecas utilizar hasta decisiones de implementación.

Interfaz

Como se menciona en la sección 1.2, uno de los requerimientos del proyecto es poder brindar al usuario una interfaz gráfica que permita la interacción con la herramienta.

Para la interfaz gráfica se eligió el módulo PyQt (en particular la versión 5) el cual es la biblioteca gráfica Qt para Python [43]. Es un *Framework* que ofrece un diseño agradable para el usuario, existe bastante documentación (mucha de la información se encuentra en lenguaje C++ aunque es fácilmente modificable a Python) así como ejemplos de su uso. Además, QT ofrece un software llamado QTdesigner [49] el cual se puede descargar de forma gratuita, que ayuda en el diseño de las herramientas y que luego se adapta fácilmente al código. Este último punto tiene una importancia significativa, ya que no se tenía experiencia en el diseño de una interfaz de usuario.

Python provee de manera nativa la biblioteca TKinter la cual es la interfaz estándar para el kit de herramientas Tk GUI (Graphical User Interface) [46]. Debido a los resultados obtenidos en pruebas primarias y ejemplos de uso evaluados, se descartó utilizar esta biblioteca ya que se entendió que la curva de aprendizaje para lograr un diseño que fuera agradable a nuestros ojos, era pronunciada.

Otra posibilidad era utilizar WxPython. Si bien no tiene los problemas de TKinter, no posee de manera nativa una herramienta que ayude en el diseño. Además, no cuenta con una amplia documentación y ejemplos como TKinter, PyQT y Kivi [59].

Por último se investigó Kivy. Está más enfocada en la parte de desarrollo de aplicaciones móviles, pero también se pueden desarrollar herramientas como la implementada en este proyecto. Es la que permite un mejor diseño y además cuenta con herramientas que facilitan el mismo. Como contrapartida, no es sencillo adaptar el código de la interfaz y la mayoría de los ejemplos son de desarrollo de aplicaciones móviles [28].

Biblioteca	Diseño Agradable para el Usuario	Documentación	Posee herramienta para Diseño	Simpleza para Adaptar
PyQT	Favorable	Muy Favorable	Si	Muy Favorable
TKinter	Un poco Favorable	Muy Favorable	No	Muy Favorable
WxPython	Favorable	Favorable	No	Muy Favorable
Kivy	Muy Favorable	Favorable	Si	Un poco Favorable

Tabla 6.1: Comparativa bibliotecas para Interfaz gráfica

En la tabla 6.1 se pueden ver los puntos que se compararon de las distintas herramientas, se clasificaron según: Muy Favorable, Favorable, Un poco Favorable, Nada Favorable y Si o No en el caso de poseer herramienta de diseño. Para definir si es agradable para el usuario, al no tener la posibilidad de hacer un estudio sobre esto, se tomó en cuenta nuestra opinión y gente allegada a nosotros.

Manejo de hilos

Al tratarse de una herramienta con alto procesamiento de información, pero a la vez plantearse la necesidad de trabajar interactivamente con el usuario, se diseñó una herramienta multihilo. De manera simplificada, se tiene un hilo principal de ejecución que es el que se encarga de la interacción con el usuario en todo momento, y otros que se crean a demanda para la carga y procesamiento de datos al efectuar alguno de los métodos. La ventaja que se obtiene con esta solución es que se evitan los bloqueos que se le presentan al usuario al seleccionar un método, ya que el hilo debe procesar esa información y mientras esto se produce, no puede atender las solicitudes efectuadas por el usuario.

Python provee de manera nativa una biblioteca para este manejo llamada Multithreading pero no es sencillo hacerlo trabajar con PyQT (el cual es el marco utilizado para desarrollar la herramienta y del que se hablará luego). Qt proporciona una interfaz muy simple para ejecutar trabajos en otros hilos. Se basa en dos clases: QRunnable y QThreadPool. El primero es el contenedor del trabajo que desea realizar, lo cual en el contexto del proyecto se mapea con los métodos de carga y procesamiento en las funciones. El segundo es el método mediante el cual el trabajo es ejecutado por hilos alternos. Cuando se ejecuta la función *start* del hilo creado por el QthreadPool, se ejecuta la función *run* definida en el QRunnable.

Es necesario poder controlar cuando un hilo termina de forma satisfactoria, y con esto recuperar el resultado en el hilo principal, así como también poder manejar los errores desde el hilo principal. Para esto Qt proporciona un marco de señales, lo que permite una comunicación segura directamente desde los subprocesos en

```
thread = plotSurface(strain,repetition, fvp, lvp) #create QRunnable plotSurface class
thread.signals.error.connect(self.errorPlot)
thread.signals.finished.connect(self.plotSurfaceOK)
self.loadingPS.setMovie(self.movie)
self.movie.start()
self.threadpool.start(thread) #start thread
```

Figura 6.1: Ejemplo inicio de hilo para el calculo de superficie de entropía

```
class plotSignals(QObject):

    finished = pyqtSignal(list)
    error = pyqtSignal(object)

class plotSurface(QRunnable): #process in other thread plot surface

    def __init__(self,strain,repetition,fvp,lvp):
        super(plotSurface, self).__init__()
        self.strain = strain
        self.repetition = repetition
        self.fvp = fvp
        self.lvp = lvp
        self.signals = plotSignals()
    @pyqtSlot()
    def run(self):
        try:
            path_pkl = 'data_experiments/'
            experimentName = self.strain + '_' + self.repetition + '(' + self.fvp + '-' + self.lvp + ')_entropy.parquet' #get experiment
            df = pd.read_parquet(path_pkl + experimentName) #read experiment
            getEntropySurface(df, self.strain, self.repetition) #calculate entropy surface
            self.signals.finished.emit([self.strain, self.repetition, self.fvp, self.lvp])  # done, send info to main thread
        except Exception as e:
            self.signals.error.emit(e)
```

Figura 6.2: PlotSurface hereda de QRunnable, la función *run* inicia al momento de la ejecución la función *start* desde el hilo principal

ejecución al módulo principal. Las señales le permiten emitir valores, que luego se recogen en otra función previamente conectada a esa señal.

En este proyecto, en el módulo principal se tiene un objeto del tipo QThreadPool, el cual es el encargado del manejo de hilos. Al momento del usuario solicitar algún método, las funciones en el módulo principal obtienen toda la información necesaria, y dependiendo del método se crea un objeto que hereda de la clase QRunnable. Éste objeto recibe los parámetros de entrada necesarios para el procesamiento solicitado, y se conecta el resultado de la ejecución a dos funciones, una para el error y otra para la ejecución satisfactoria. Al finalizar se ejecuta una de estas funciones que es la que despliega el resultado al usuario. En las imágenes 6.1 y 6.2 se puede observar este comportamiento.

6.3.2. Capa de negocio

En esta sección se detallarán la estructura utilizada para trabajar con los datos en memoria, el flujo de trabajo entre las tres capas a cargo del módulo principal, así como también los métodos implementados.

	Position	Coverage	Strain	Repetition	Passage	Entropy	info
257	774	16657	299	1	1	0.022749	299,1,1,774
258	777	16219	299	1	1	0.024971	299,1,1,777
259	780	16551	299	1	1	0.018132	299,1,1,780
260	783	17405	299	1	1	0.024274	299,1,1,783
261	786	16034	299	1	1	0.038979	299,1,1,786
...	...	...	...	...	...	...	...
2439	7320	2423	299	1	40	0.027640	299,1,40,7320
2440	7323	2358	299	1	40	0.039383	299,1,40,7323
2441	7326	2556	299	1	40	0.035478	299,1,40,7326
2442	7329	2475	299	1	40	0.065377	299,1,40,7329
2443	7332	2420	299	1	40	0.015875	299,1,40,7332

[45927 rows x 7 columns]

Figura 6.3: DataFrame Cepa 299, repetición 1, FVP 774, LVP 7332, con entropía

Manejo de datos en memoria

En una primera instancia se estudió qué estructura utilizar al momento de trabajar con la información. Esta decisión era importante de cara al diseño y desarrollo de la herramienta. En el proyecto que antecede, se utilizó mayoritariamente para el manejo de la información la biblioteca Numpy.

Se investigó qué bibliotecas se utilizan mayoritariamente en el área de ciencia de datos para manipular y almacenar en memoria la información. En esto se destaca por su amplia utilización la biblioteca Pandas y en particular la estructura DataFrame.

Pandas es una biblioteca de Python que se utiliza principalmente para la manipulación y el análisis de datos. Posee tablas (denominadas DataFrames), filas y columnas (series), y muchas otras funcionalidades (tales como la posibilidad de generar gráficas, guardar y cargar información, cálculos matemáticos, entre otros). Esto la convierte en una biblioteca poderosa para el procesamiento y manipulación de datos [61].

Pandas trabaja de manera nativa con otras bibliotecas tales como Numpy y Matplotlib [33]. Estas últimas se utilizan en el proyecto anterior y por lo tanto en este se continuó haciendo uso de las mismas.

En la figura 6.3 se puede observar de qué manera se ve un experimento almacenado utilizando DataFrame de Pandas. En el proyecto se utiliza la versión de Pandas 1.0.3 publicada en Marzo del 2020.

Módulo Principal

El módulo principal es el que se encarga de controlar y manejar el flujo de todas las funcionalidades, los experimentos activos cargados, el manejo de los hilos así como también el registro y la comunicación con la base de datos.

Este módulo crea una clase llamada *Interface* la cual hereda de QMainWindow perteneciente a PyQT5. Esta clase proporciona un marco para crear la interfaz de usuario de una aplicación. QT cuenta, entre otras, con QMainWindow para la gestión de la ventana principal [48].

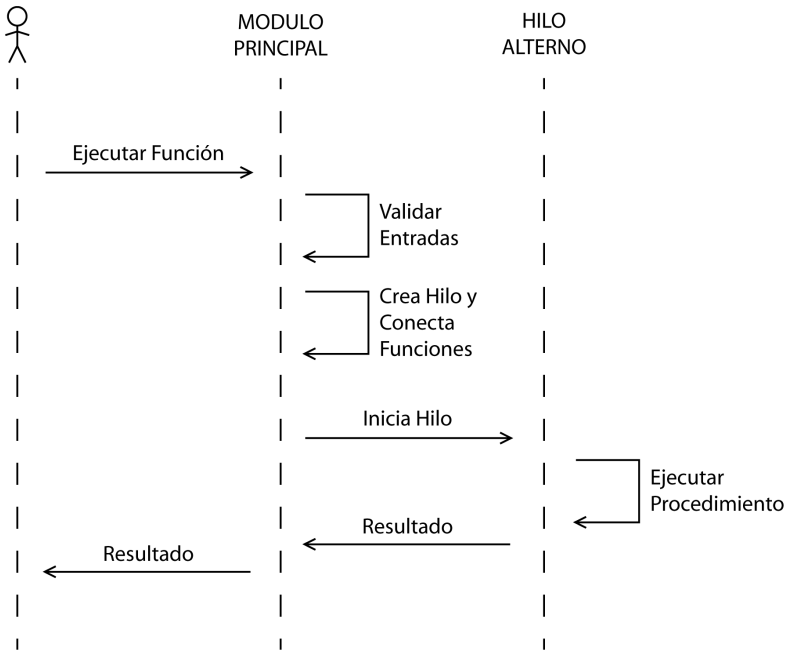


Figura 6.4: Diagrama de flujo de ejecución de funciones por el usuario

Carga un archivo UI el cual tiene un formato XML y contiene todo el diseño de la herramienta desarrollado utilizando QTDesigner. Este archivo contiene toda la información de los distintos botones, etiquetas, widgets, posicionamiento de los mismos y nombres, entre otros.

Como se observa en la figura 6.4 es el encargado de interactuar con el usuario, por lo cual tiene mínimamente una función por cada acción que el usuario puede realizar. Efectúa el control de los parámetros ingresados por el usuario para así realizar el manejo y control de errores. Este módulo es quien presenta los resultados al usuario.

Métodos de carga y cómputo

A partir de los requerimientos funcionales presentados en la sección 6.2 se desarrollaron los distintos métodos que se irán detallando a continuación.

Carga de directorios

Para la carga de directorios, una vez que el usuario selecciona la carpeta (la cual contiene un conjunto de archivos CSV donde cada uno de ellos se corresponde con un pasaje para una determinada cepa y repetición) y define las posiciones válidas se crea un hilo para el procesamiento de la información. En una primera instancia,

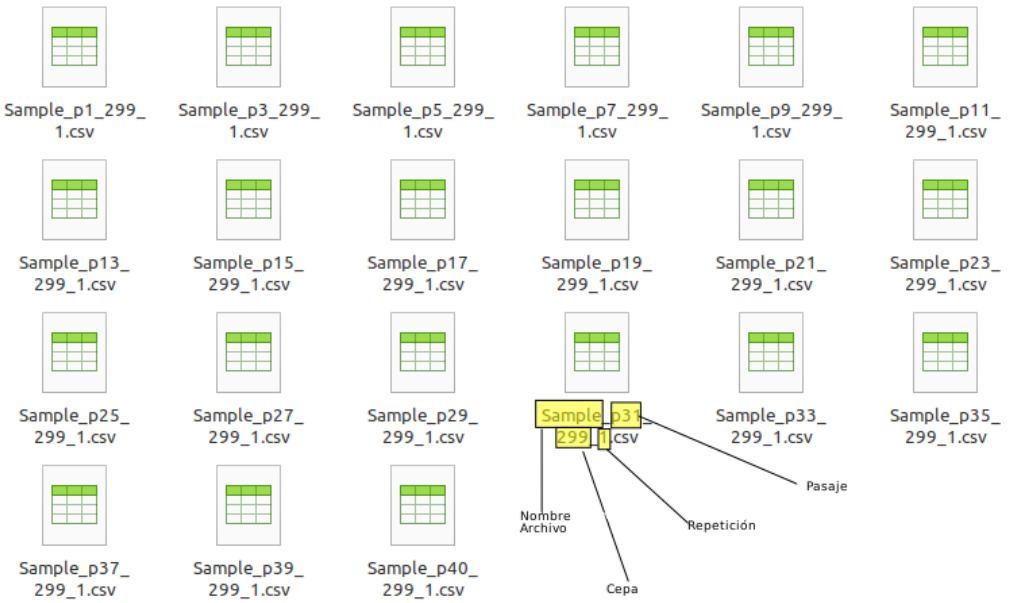


Figura 6.5: Directorio con Archivos CSV

Ejemplo de formato de nombre del archivo CSV admitido por la herramienta

la función `readData`, lista todos los archivos CSV de la carpeta y crea un DataFrame con las frecuencias relativas de los codones y la información del experimento. Los archivos CSV contenidos en el directorio deben ajustarse al formato establecido en la figura 6.5. Puesto que cada archivo CSV tiene n filas correspondientes a cada posición del genoma secuenciado, y el directorio contendrá p archivos correspondientes a cada uno de los pasajes, el DataFrame resultante tiene $n \times p$ filas. Si alguno de los archivos CSV no cumple con los requisitos establecidos, se notificará al usuario con un mensaje de error. En caso de que los parámetros ingresados por el usuario y el formato de los archivos contenidos en el directorio sean válidos, se guarda un DataFrame en formato Parquet [39] y se pasa a procesar la entropía utilizando la función `getEntropy`. A partir de la frecuencia relativa de los codones para cada dupla $(pasaje, posición)$ se calcula la entropía utilizando la ecuación 3.2 presentada en el capítulo 3. Una vez calculada la entropía se descarta la información de los codones (ya almacenada en el archivo Parquet previamente generado por `readData`) y se guarda el nuevo DataFrame. Nuevamente, éste DataFrame contiene $n \times p$ filas.

Carga de archivos

Puesto que la herramienta guarda en disco un archivo por cada Experimento procesado, se da la posibilidad de cargarlo evitando computarlo nuevamente. El archivo en formato Parquet es seleccionado y de esta manera el experimento queda disponible para que el usuario pueda trabajar con el mismo. Como se observa en la figura 6.6 es posible únicamente cargar archivos con extensión .parquet

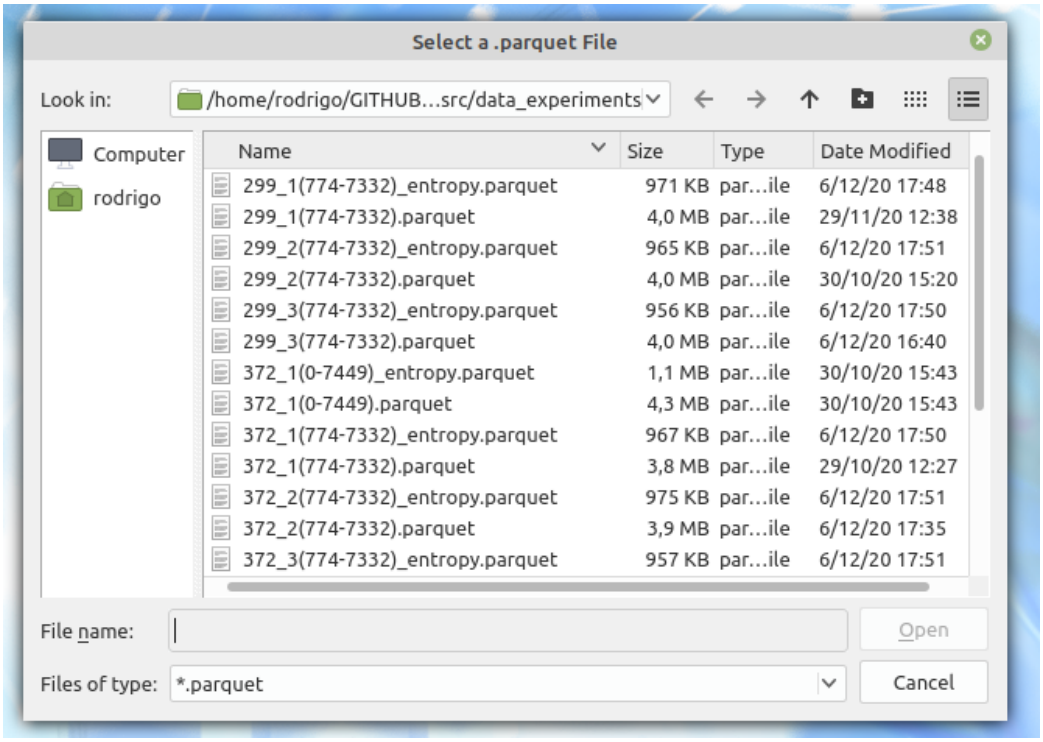


Figura 6.6: Diálogo para la carga de experimentos ya procesados

Entropía por Posición

A partir de un experimento y una posición, éste método se encarga de mostrar la evolución de la entropía en el tiempo. A partir del Dataframe del experimento, se toma la columna de entropías para esa posición en todos los pasajes y esto se grafica utilizando Matplotlib. El resultado queda almacenado de forma tal que el usuario lo pueda consultar en otro momento.

En la figura 6.7 se puede observar como se visualiza esta función para un experimento y una posición

Superficie Entropía

Este método, se encarga de mostrar una visualización de la entropía de un experimento para todo pasaje y posición. A partir del DataFrame, se toma las columnas entropía, posición, y pasaje. Para cada una de éstas se efectúa un redimensionamiento de manera de obtener tres arreglos de numpy con dimensión dos, y cuya nueva dimensión es $(pasajes, posiciones)$. Así, la coordenada x corresponde a los pasajes, la coordenada y a las posiciones, y z a la entropía. Con estos nuevos vectores, se grafica utilizando la función `contour3D` de Matplotlib. Al igual que la Entropía por Posición, este es un método estático y el resultado queda almacenado.

En la figura 6.8 se puede observar cómo se visualiza esta función para un experimento

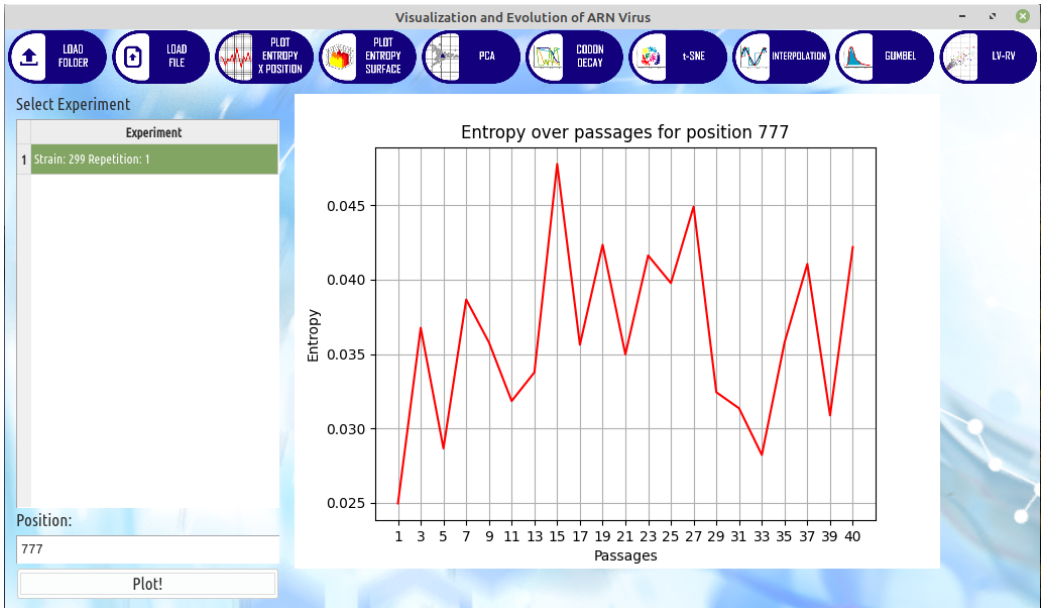


Figura 6.7: Entropía por Posición

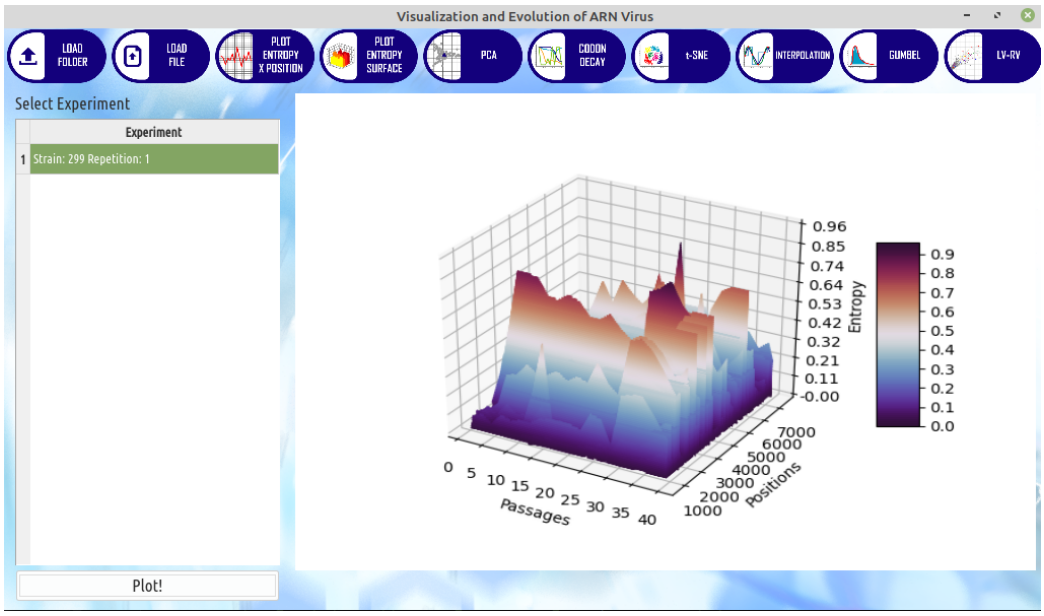


Figura 6.8: Superficie Entropía para cepa 299 repetición 1

Análisis de máximos de entropía - Gumbel

La funcionalidad de la herramienta *Gumbel* muestra otro tipo de visualización estática. La técnica teórica se detalla en mayor profundidad en la sección 3.3.

A partir del experimento seleccionado por el usuario, se obtiene el DataFrame correspondiente al mismo. Con éste se obtienen los máximos de entropía para cada posición. A partir de los máximos obtenidos, se utiliza la función de *Pandas*

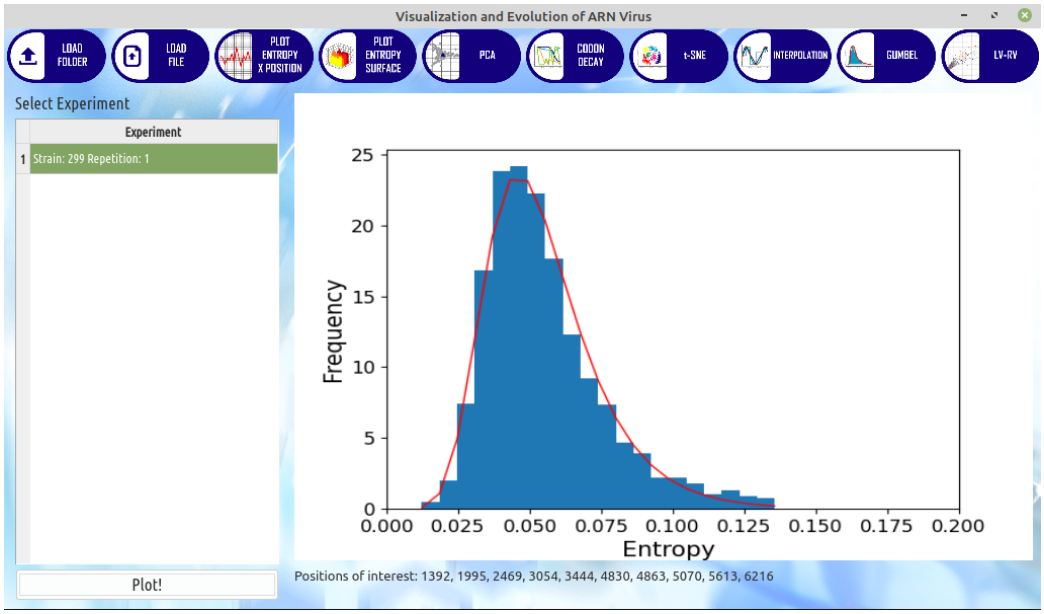


Figura 6.9: Análisis de máximos de entropía para experimento con cepa 299, repetición 1, FVP 774 y LVP 7732. Se puede visualizar, por debajo de la gráfica las diez posiciones con menor $p - valor$

`quantile` con parámetro 0,95 de forma obtener el valor que separa la distribución en 95 % y 5 %. La cola por derecha (es decir, los datos mas extremos) del cuantil se descarta y a partir de esta distribución se genera un histograma utilizando la función `hist` de *Pandas*. Se aproxima la distribución generada anteriormente utilizando la función `fit` de *scipy* con Gumbel de forma de obtener μ y σ. Con estos datos, mas el histograma, se grafica la función de densidad de Gumbel.

Se calcula el $p - valor$ para todos los puntos incluidos los anteriormente descartados. Los diez $p - valores$ más pequeños, es decir, los datos mas extremos, se listan como posiciones de interés para el usuario.

En la figura 6.9 se puede observar como se visualiza este método para un experimento

Interpolación

Al tratarse de datos que son tomados luego de una secuenciación, éstos pueden presentar errores o datos fuera de escala. De forma de poder eliminar esta información y obtener métodos de visualización y análisis adecuados se implementaron dos tipos de interpolación para los experimentos como se observa en la figura 6.10

- Interpolación de entropía: De acuerdo al pasaje definido por el usuario, se efectúa una interpolación lineal para cada posición utilizando la información del pasaje anterior y posterior. Para esto a partir del DataFrame del experimento seleccionado, se obtiene la información del pasaje anterior y

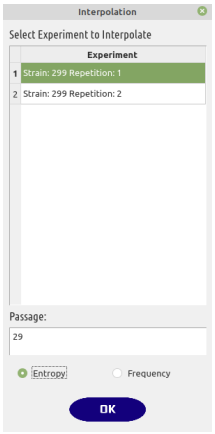


Figura 6.10: Interpolación de entropía para experimento con cepa 299,repetición 1, FVP 774, LVP 7332 en pasaje 29

posterior y se aplica la interpolación lineal utilizando la función de *Pandas* `interpolate`

- Interpolación de frecuencias: Al igual que la interpolación por entropía, el usuario elige el pasaje y se efectúa una interpolación lineal para cada dupla $(posición, codón)$. Se implementa este método debido a que existen funciones que no utilizan la entropía sino la frecuencia relativa de los codones. Al igual que la interpolación de entropía, se utiliza la función `interpolate`.

Esta interpolación modifica los datos del experimento para luego almacenarlo. Esto quiere decir que si el usuario vuelve a cargar la información del experimento, si se efectuó una interpolación en la anterior sesión, la misma información contenida en el experimento ya estará interpolada.

Análisis mediante variación de frecuencia de codones dominantes - Codon Decay

El primero de los métodos interactivos implementados es Codon Decay. Como se menciona en el capítulo 3, a diferencia de los otros métodos, éste utiliza la frecuencia relativa de los codones en vez de la entropía.

A partir del experimento seleccionado, se obtiene el DataFrame del mismo. Con éste se obtiene para cada posición el codón más frecuente y se crea un arreglo de Numpy. Se utiliza la función `amin` de Numpy para obtener el valor mínimo del arreglo. Si éste se encuentra por debajo del umbral seleccionado por el usuario, se grafica la evolución de esta dupla $(codón, posición)$ en los distintos pasajes utilizando Matplotlib.

De forma de facilitar la visualización (ya que en caso de elegir umbrales altos, la cantidad de codones podría ser alta), la información relativa a la dupla $(codón, posición)$ se muestra cuando el usuario pone el cursor del ratón sobre la gráfica de la cual quiere obtener información. En la figura 6.11 se puede ver como se comporta el método.

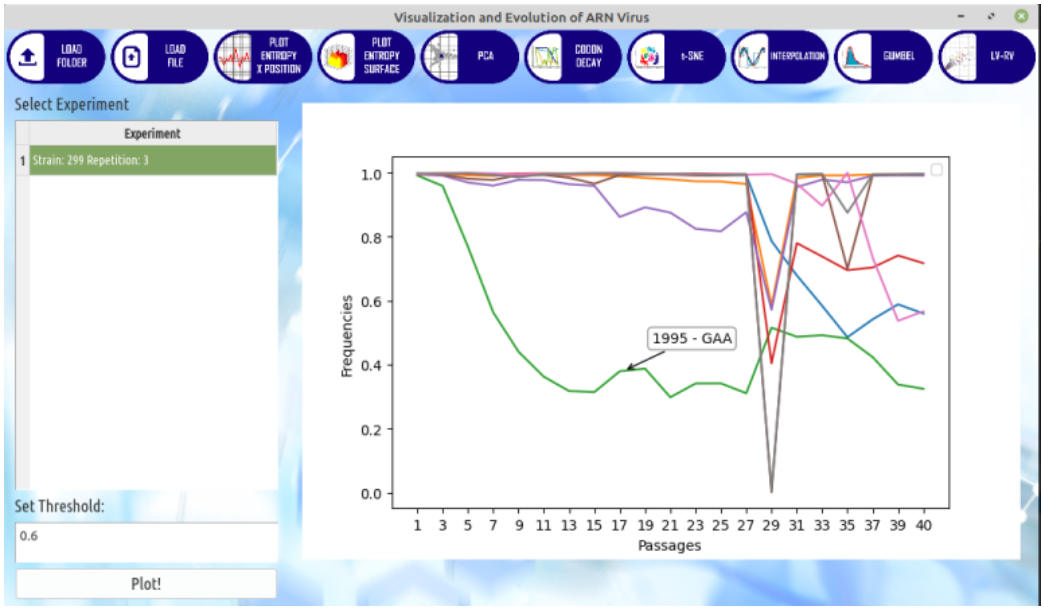


Figura 6.11: Análisis mediante variación de frecuencia para cepa 299, repetición 3, FVP 774, LVP 7332 con umbral 0,6

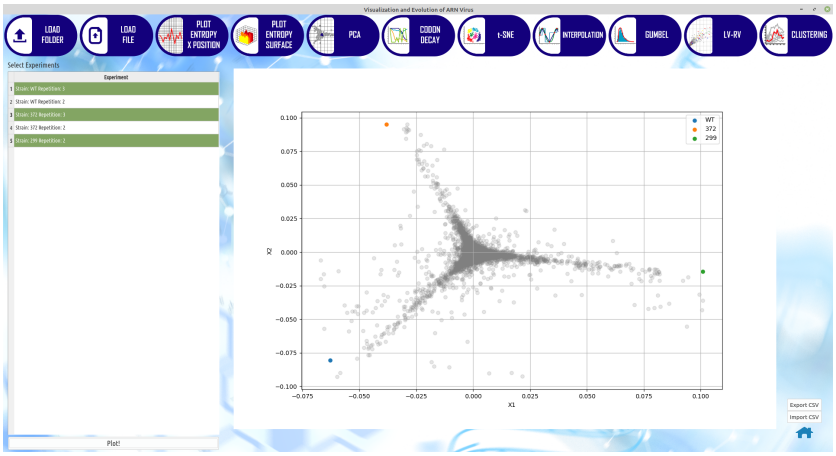
PCA

Como se menciona anteriormente en la sección 3.4, PCA es una transformación lineal que escoge un nuevo sistema de coordenadas para el conjunto original de datos. La varianza de mayor tamaño del conjunto de datos es capturada en el primer eje (llamado el Primer Componente Principal), la segunda varianza más grande es el segundo eje, y así sucesivamente. En nuestro proyecto, se utilizó PCA para bajar la dimensión de los datos a dos coordenadas.

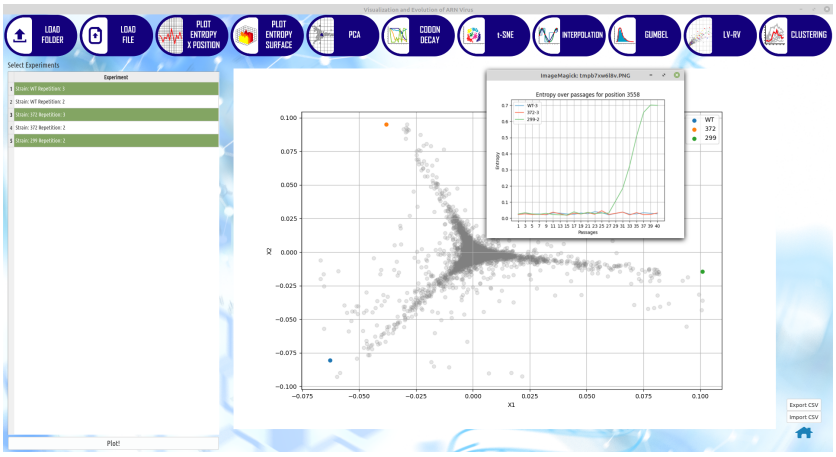
A partir de los experimentos seleccionados por el usuario, se alinean las posiciones. Este paso se realiza debido a que los experimentos podrían contener distintos FVP y LVP y de esta forma, las posiciones contenidas serían distintas. Posteriormente, se aplica a la entropía de cada experimento la transformación PCA. Es decir se aplica PCA a un vector de dimensión $Pasajes \times Posiciones$. Se utiliza la función `PCA` provista por la biblioteca scikit-learn. Como resultado se obtiene la misma cantidad de puntos que experimentos seleccionados. Asimismo, se obtienen los vectores propios de la transformación.

Con la información obtenida, se grafican los puntos en los nuevos ejes de coordenadas y los vectores propios. Considerando que estos podrían estar en distintas escalas, se le aplica una homotecia a los nuevos puntos utilizando información de los promedios. De esta manera se logra una mejor visualización de la influencia de los puntos en la distribución. En la figura 6.12 se puede observar el método.

El usuario tiene la posibilidad de manera interactiva (haciendo un click) de ver como a partir de un punto del vector, se obtiene la dupla $(posición, pasaje)$ del índice correspondiente a ese punto. A partir de eso se grafica la entropía por posición para todos los experimentos. Para el caso de los datos con los que se trabajó ésta visualización muestra cómo la trayectoria de entropía en ciertas posiciones



(a)



(b)

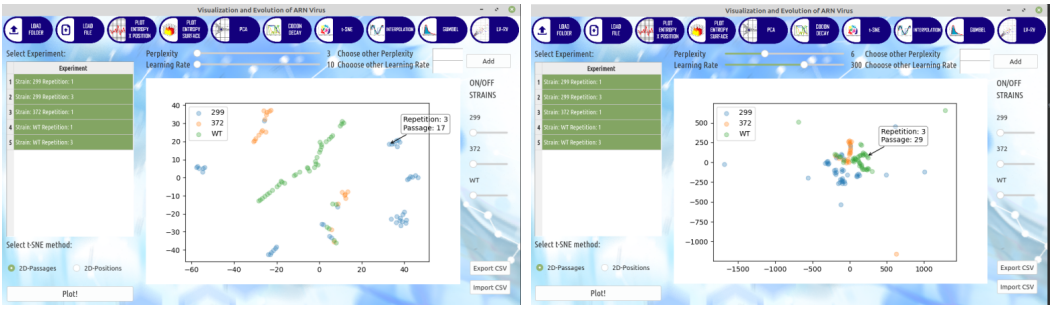
Figura 6.12: Descomposición PCA para tres experimentos con cepas distintas y trayectoria de entropía para posición 3558 con Entropía por posiciones

separa las distintas cepas.

Por último, este método provee la capacidad de exportar los resultados para en caso de necesitarlo, volver a graficar la información y/o obtener los valores de los puntos en las nuevas coordenadas para posterior análisis. Se exporta un archivo en formato CSV que contiene toda la información necesaria para visualizar la técnica.

La importación recibe un CSV válido (es decir, el resultado de un PCA ya procesado) y despliega nuevamente la información de manera visual. Al no contar con la información de los experimentos (ya que en este caso no es necesario cargar los experimentos) no se puede utilizar la funcionalidad de graficar la entropía por posición al hacer un click.

Visualización t-SNE



(a) LR=10 y Perplexity=3 (b) LR=300 y Perplexity=6

Figura 6.13: t-SNE aplicado a los pasajes, utilizando cinco experimentos.

El método t-SNE, el cual se detalla en mayor profundidad en el capítulo 4 se implementa en dos tipos, t-SNE aplicado a los pasajes y t-SNE aplicado a las posiciones. Esto es que para t-SNE por pasajes, tenemos P puntos, con P= cantidad de pasajes donde cada punto tiene dimensión n, con n=cantidad de posiciones. Para el caso t-SNE aplicado a las posiciones, tenemos n puntos con dimensión P.

Al igual que en PCA, previamente a la ejecución, se alinean las posiciones y además se reorganiza la información contenida en el DataFrame, para obtener un arreglo donde cada fila es la trayectoria de entropía para una posición p. Se utiliza la función `TSNE` provista por la biblioteca scikit-learn. Esta función, además de la información de los experimentos, requiere dos parámetros: learning rate (LR) y perplexity.

Por defecto se calcula con las combinaciones de LR $= [10, 100, 300, 500]$ y Perplexity $= [3, 6, 12, 15]$ para t-SNE por pasajes. Para t-SNE por posiciones LR=$[10, 300]$ y Perplexity $= [3, 12]$. Se efectúan menos visualizaciones para las posiciones debido a que la cantidad de puntos a calcular, aumenta el tiempo de procesamiento. Las distintas visualizaciones obtenidas se pueden desplegar a demanda moviendo las barras provistas para esto. En la figura 6.13 se puede observar como se visualizan éstas funcionalidades.

En caso que el usuario desee ingresar parámetros distintos a los por defecto, se provee una funcionalidad donde puede elegir los parámetros que entienda necesario mediante el botón `Add`. Para esto, se vuelve a procesar el método con los nuevos parámetros seleccionados.

De forma de facilitar la visualización del usuario, se provee la funcionalidad que oculta los puntos pertenecientes a las distintas cepas. Para esto, el usuario utiliza las barras provistas a la derecha en la herramienta, las cuales se cargan dinámicamente a partir de los experimentos seleccionados. En la figura 6.14 se puede observar este comportamiento.

De forma de poder obtener información de los puntos, cuando el usuario coloca el cursor por encima de uno de ellos, se muestra la información correspondiente al mismo.

Por último este método provee la capacidad de exportar los resultados para en caso de ser necesario, volver a graficar y/o obtener la información del valor que le

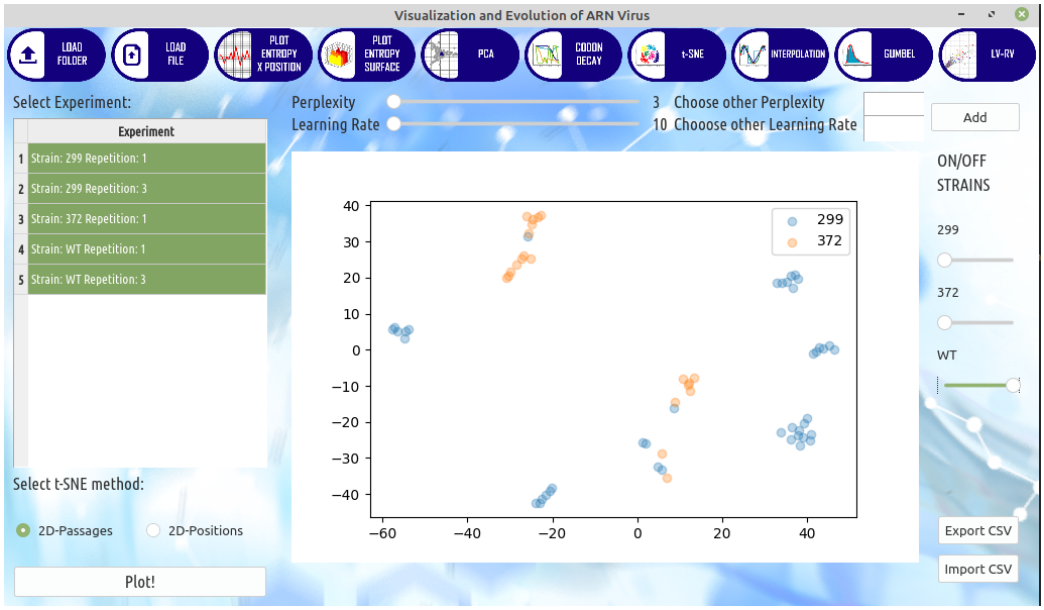


Figura 6.14: t-SNE aplicado a pasajes, con parámetros LR=10 y Perplexity=3, filtrando la información de los puntos de la cepa WT

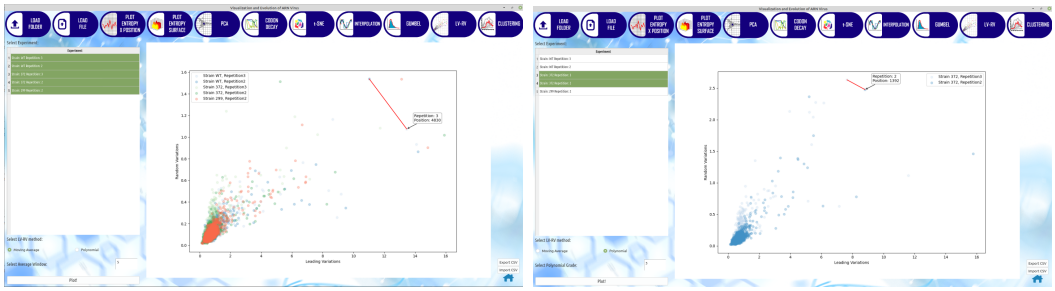
asigna a las distintas posiciones este algoritmo para posterior análisis. Se exporta un archivo en formato CSV que contiene la información necesaria para visualizar la técnica.

La importación recibe un CSV válido (es decir, el resultado de un t-SNE ya procesado) y muestra nuevamente la información de manera visual. Todas las funcionalidades anteriormente descriptas funcionan, salvo únicamente la opción de agregar una dupla $(Perplexity, LR)$, debido a que no se cuenta con la información de los experimentos.

LV-RV

Leading Variations - Random Variations(LV-RV) muestra visualmente el análisis de los datos descripto en la sección 3.2.

De igual manera a los métodos anteriores, previamente a la ejecución, se alinean las posiciones y se reorganiza la información contenida en el DataFrame. Para la aproximación por polinomio, se utiliza la función `polyfit` de Numpy, la cual aproxima la trayectoria de entropía para cada posición con un polinomio de grado g seleccionado por el usuario. Retorna un vector de coeficientes que minimiza el error utilizando mínimos cuadrados. Con estos puntos resultantes se calcula el Leading Variations el cual es el acumulado de la suma de los puntos del polinomio y el Random Variations que es el acumulado de la resta de la entropía con el aproximado por el polinomio. Se crea un nuevo DataFrame con esta información y se grafica utilizando Matplotlib. Para LV-RV por promedio móvil, para cada punto de la trayectoria de entropía se calcula el promedio móvil utilizando la ventana escogida por el usuario. Al igual que en la aproximación por polinomio, se calcula



(a) Promedio móvil (b) Polinomio

Figura 6.15: Aplicación de LV-RV por ambas aproximaciones, utilizando como parámetros cinco para la ventana deslizante y tres para el grado del polinomio

el acumulado y la diferencia, se crea un nuevo DataFrame y se grafica utilizando Matplotlib.

Al seleccionar un punto con el puntero del ratón, se muestra la información del mismo así como también, se conectan puntos de la misma cepa e igual posición . Esto último permite ver como se comporta esa posición. En la figura 6.15 se puede observar el funcionamiento del método.

Al igual que los métodos anteriores se provee la posibilidad de exportar la información e importar. Al importar un resultado, todas las funcionalidades descriptas anteriormente están disponibles.

Agrupación de series temporales - Clustering

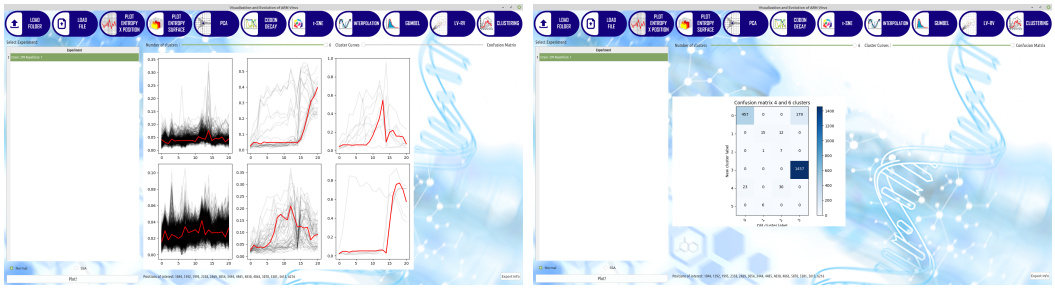
El método de agrupación de series temporales fue una de las nuevas técnicas desarrolladas en el proyecto para detectar posiciones de interés. La técnica se explica en mas detalle en el capitulo 5, aquí solo nos vamos a referir a la implementación del método.

Se desarrollaron dos técnicas:

- Agrupación de las trayectorias de entropía reales
- Agrupación utilizando la aproximación con la técnica Singular Spectrum Analysis (SSA)

A partir del DataFrame del experimento seleccionado, se reorganiza la información contenida en el DataFrame. Utilizando la biblioteca `TimeSeriesKMeans` de Tslearn, se realiza la agrupación para dos, cuatro y seis grupos de las trayectorias de entropía. Como se explicó en el capítulo 5, se utilizó la métrica DTW (Dynamic Time Warping). Para la agrupación SSA, se realiza la descomposición utilizando la función `SingularSpectrumAnalysis` provista en la biblioteca Pyts y se realiza la agrupación pasando como serie temporal, la primer componente de la descomposición.

A partir de las etiquetas asignadas a las trayectorias, para las distintas agrupaciones de dos, cuatro y seis grupos, se realiza la matriz de confusión de forma de visualizar como se descompone cada agrupación.



(a) Trayectorias (b) Matriz de confusión

Figura 6.16: Aplicación del método Clustering con valores reales, para cepa 299, repetición 1, FVP 774, LVP 7332 utilizando seis grupos

El usuario puede seleccionar qué visualización desplegar, ya sea la matriz de confusión o trayectorias, así como también seleccionar la cantidad de grupos.

Por último, se muestran posibles posiciones de interés para el usuario. Estas posiciones surgen de la agrupación con seis grupos. Para cada uno de estos últimos se contabiliza cuántas trayectorias tiene asignadas. Si el cardinal del grupo es menor igual que el 1 % de la cantidad total de posiciones del experimento, se guardan las posiciones y se muestran al usuario. En la figura 6.16 se puede observar las distintas visualizaciones del método.

Por último, todas las visualizaciones quedan almacenadas para posterior análisis. Además, se puede exportar la información de las etiquetas asignadas a cada posición mediante la funcionalidad de exportar, la cual retorna un archivo CSV.

Exportar resultados obtenidos

Si bien no es un método analítico, la herramienta provee la posibilidad de exportar los resultados de los experimentos que se han efectuado. Para esto la herramienta tiene que estar en modo en línea ya que esta consulta se hace a la base de datos. Se exporta un archivo CSV con toda la información referente a los resultados de los análisis efectuados.

6.3.3. Capa de persistencia

En esta sección se detalla de qué manera se almacena la información en disco y cómo se registra y recupera la actividad de usuario. Además se exhiben comparativas utilizadas con el fin de tomar decisiones referentes al diseño.

Estructura de almacenamiento

Para optimizar tiempo y recursos, así como también no mantener información en memoria, para este proyecto, se tiene como requerimiento guardar la información ya procesada y cargarla en memoria a demanda.

Pandas, provee métodos para guardar la información en distintos formatos, algunos de los mas importantes son: [60]

- **CSV:** El formato mas típico, y como vienen los archivos de los experimentos antes de ser procesados
- **Pickle:** Formato que utiliza Python para serializar los datos
- **MessagePack:** Formato parecido a JSON, rápido y pequeño
- **HDF:** Formato de archivo diseñado para almacenar y organizar grandes cantidades de datos
- **Feather:** Un formato binario, el cual es rápido, liviano y simple de utilizar
- **Parquet:** Formato de almacenamiento de Apache Hadoop (*Framework* que permite el procesamiento distribuido de grandes conjuntos de datos en grupos de computadoras utilizando modelos de programación simples [4])

De estos últimos, se profundizó el estudio en los siguientes:

- **Pickle:** Debido a que es nativo de Python, por lo cual la adaptabilidad al código es sencilla, tiene mucha documentación y en las pruebas realizadas, se obtuvieron buenos resultados.
- **Feather:** Formato binario el cual es rápido y liviano y que se puede intercambiar con el lenguaje R, utilizado en el área de la ciencia de datos.
- **Parquet:** Al ser utilizado en Hadoop, existe mucha información y al trabajar con grandes volúmenes de datos, optimiza bien el tamaño.
- **HDF:** Al igual que Parquet, se utiliza para trabajar con grandes volúmenes de datos lo cual podría hacerlo adecuado a la posibilidad de escalar este proyecto.

Además, se realizaron pruebas de forma de comparar el funcionamiento de éstos y de esta manera, tomar la decisión de que estructura utilizar.

Antes de hablar de los distintos tipos es importante mencionar a qué se hace referencia cuando se habla de serialización y deserialización. **Serialización** es un mecanismo para convertir el estado de un objeto en una secuencia de bytes. **Deserialización** es el proceso inverso donde el flujo de bytes se usa para recrear el objeto real en la memoria.

Pickle

Pickle implementa protocolos binarios para serializar y deserializar una estructura de objeto Python. El proceso de serialización o "Pickling" es en el cual el objeto Python se convierte en una secuencia de bytes y mediante la función `dump` se puede guardar en un archivo. El proceso de deserialización o "Unpickling" es

la operación inversa, mediante la cual la anterior secuencia de bytes se convierte nuevamente en el objeto [44]. Para esto último se utiliza la función `load`.

Si bien Pickle posee muchas características positivas para su uso en el proyecto, posee un defecto importante, vinculado a su trabajo con Pandas y la función `read_pickle`. Esta función es la que deserializa el Pickle y vuelve a transformar esos bytes en el DataFrame anterior. Si las versiones de Pickle y Pandas son distintas (es decir, ese objeto Pickle fue guardado con una versión mas antigua de Pandas), produce un error: no puede deserializar correctamente [56]. Esto podría ocasionar problemas al momento de compartir información de experimentos que ya fueron procesados.

Feather

Feather es un formato de archivo para almacenar DataFrames (de lenguajes como Python o R) que utiliza el formato Arrow IPC internamente. Feather surge como una prueba de concepto para el almacenamiento rápido de DataFrames independientes del lenguaje [15].

Uno de los puntos fuertes de Feather es la posibilidad de intercambiar datos entre R y Python los cuales son lenguajes ampliamente utilizados en el área de la ciencia de datos. Feather es un formato de archivo binario rápido, liviano, independiente del lenguaje y fácil de usar. Proporciona serialización en columnas binarias, diseñada para hacer una lectura y escritura eficiente de los DataFrames [9].

Si bien tiene muchas características positivas, no está carente de limitaciones. Algunas de las mismas al utilizarla con Pandas son [16]:

- No soporta nombres de columnas que no sean tipo *string*
- No soporta índices en las filas
- No soporta columnas con datos no homogéneos

Como está todavía en continuo desarrollo (no existe una versión estable), no se recomienda que se utilice para almacenar información por mucho tiempo. En nuestro contexto, es importante que la información del experimento se pueda reutilizar sin importar el tiempo que pase [17].

Parquet

Parquet es un formato de archivo de código abierto diseñado por Apache para utilizar en el ecosistema de Hadoop. Está diseñado en un formato eficiente en el almacenamiento de datos, utilizando columnas planas [3].

Parquet está optimizado para trabajar con datos complejos en masa y presenta diferentes formas de compresión de datos y tipos de codificación eficientes [3].

Trabaja por columnas, lo que permite utilizar diferentes esquemas de codificación para texto y datos enteros [3].

De forma de optimizar su funcionamiento, utiliza RLE (Run-Length Encoding), que almacena un solo valor una vez junto con el número de ocurrencias. [3]:

HDF

Formato de datos jerárquico (HDF) es un formato de archivo de código abierto que admite grandes volúmenes de datos, complejos y heterogéneos. HDF utiliza una estructura similar a un directorio de archivos como puede ser un directorio de linux. Esto le permite organizar los datos dentro del archivo de muchas formas estructuradas diferentes, como lo haría con los archivos en una computadora [24].

El modelo HDF presenta dos tipos de objetos importantes [24]:

- Grupo: un elemento similar a una carpeta dentro de un archivo HDF que puede contener otros grupos o conjuntos de datos dentro de él.
- Conjunto de datos: los datos reales contenidos en el archivo HDF. Los conjuntos de datos se almacenan normalmente dentro de grupos en el archivo.

Para realizar las pruebas se utilizó la versión 5 de HDF.

Pruebas y elección

Se hicieron pruebas utilizando para ello nueve experimentos, los cuales contenían las mismas posiciones, misma cantidad de pasajes pero distinta dupla $(cepa, repetición)$. Las comparaciones realizadas entre los distintos formatos fueron:

- tiempo en segundos que tarda en guardar en disco.
- tiempo en segundos que tarda en cargar a memoria desde disco.
- tamaño en megabytes que ocupa en disco.
- tamaño en mebibytes que ocupa en memoria RAM.

El promedio de los resultados se pueden ver en las tablas 6.2, 6.3 y 6.4. Como se utilizó el promedio como medida comparativa, se calcula la desviación estándar utilizando la formula 6.1 de forma de poder determinar cuán dispersos son los datos. Una desviación baja indica que los valores de la muestra se encuentran próxima al promedio, al contrario, una desviación alta, indica que la muestra se extiende en un amplio rango de valores.

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \bar{x})^2}{N}} \tag{6.1}$$

Pickle se comporta correctamente en las distintas comparativas y es nativo de Python, presenta un problema de versionado en su trabajo con Pandas por lo que se descartó su utilización. Con respecto a Feather, al igual que Pickle, se comporta mejor en general que Parquet y HDF, pero al tratarse de un formato sin versión estable, no se recomienda su utilización para almacenar información por largo tiempo. Por ello se descartó su utilización. Parquet es el formato que

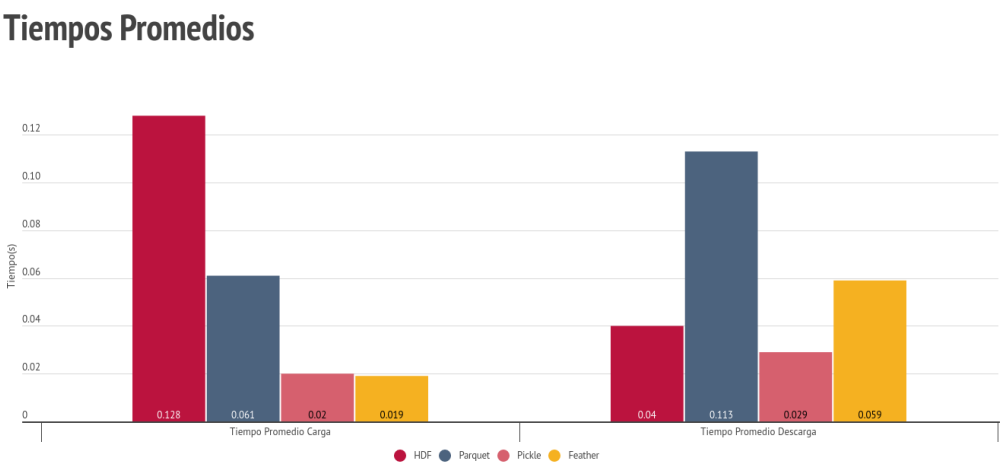


Figura 6.17: Comparativa Tiempos Formatos

	Carga	Desviación Carga	Descarga	Desviación Descarga
Pickle	0.02	0.005	0.029	0.001
Parquet	0.061	0.02	0.113	0.01
HDF	0.128	0.03	0.04	0.01
Feather	0.019	0,001	0.059	0.004

Tabla 6.2: Comparativa tiempos promedios en segundos para la carga y descarga de experimentos

presenta menor tamaño en disco, muy alejado de los valores presentes en los otros formatos. Presenta un menor tiempo promedio en la carga con respecto a HDF, pero descarga es mayor. Esto último podría deberse a alta compresión de los datos que efectúa Parquet.

Se tomó la decisión de utilizar Parquet ya que además de los resultados expuestos, es un formato independiente del lenguaje y pensando en un futuro trabajar con mayor volumen de datos, cobra mayor importancia la gran compresión que efectúa en los mismos.

	Promedio Consumo (MiB)	Desviación
Pickle	30.951	9.118
Parquet	34.442	8.468
HDF	28.113	0.864
Feather	25.743	2.510

Tabla 6.3: Comparativa consumo en memoria RAM de experimentos en Mebibyte con distintas estructuras

	Promedio Tamaño en Disco (MB)	Desviación
Pickle	25	0
Parquet	3.95	0.13
HDF	29	0
Feather	25	0.1

Tabla 6.4: Comparativa tamaño de experimentos en disco calculado en Megabyte con las distintas estructuras

```
2020-10-31 16:04:39,109 VEV Started
2020-10-31 16:05:08,762 Experiment WT_1(774-7332)was loaded and entropy was calculated with First Value Position(FVP) = 774 and Last Value Position(LVP) = 7332
2020-10-31 16:05:11,268  Experiment already calculated was load: 299_1(774-7332)
2020-10-31 16:05:32,036 Entropy is plotted and saved for position 1995 with experiment WT_1(774-7332)
2020-10-31 16:05:43,944 Gumbel is plotted and saved with experiment WT_1(774-7332)
2020-10-31 16:06:00,416 No handles with labels found to put in legend.
2020-10-31 16:06:00,457 Codon Decay was plotted with threshold 0.7 with experiment WT_1(774-7332)
2020-10-31 16:06:47,964 VEV Started
2020-10-31 16:06:58,879  Experiment already calculated was load: 372_1(0-7449)
2020-10-31 16:07:02,820 Entropy surface is plotted and saved with experiment 372_1(0-7449)
```

Figura 6.18: Ejemplo de Registro de actividad diario

Base de datos y registros

Para almacenar la información de la herramienta, y para entregarle información de los análisis ejecutados al usuario se crearon dos funcionalidades:

- Registro diario que almacena la actividad en la herramienta. Esto incluye los análisis, experimentos cargados así como también los errores.
- Una base de datos que almacena toda la información referente a los análisis y resultados obtenidos por el usuario.

Para el registro diario se utilizó una biblioteca propia de Python llamada Logging que registra toda la información de la sesión, generando un archivo diario. Aunque en los registros se almacena toda la información de actividad, debido a que la herramienta debía brindar al usuario la posibilidad de consultar los últimos experimentos en los que está trabajando, se tomó la decisión de utilizar una base de datos.

Las bases de datos basadas en documentos permiten almacenar información no estructurada. Como en un inicio, no se contaba con un esquema fijo y pensando que en un futuro, la herramienta podría evolucionar y por ende, modificar las necesidades, se tomó la decisión de utilizar una base de datos basada en documentos. En particular MongoDB [23]. Además, desde el punto de vista del desarrollo, facilita su uso ya que utiliza el mismo formato de modelo de documentos que el que se emplea en el código.

MongoDB es una base de datos distribuida, basada en documentos y de esquema libre. Provee dos opciones de uso, descargar y mantener localmente la base de datos, y usar el servicio Atlas en la nube. Éste último es gratuito hasta cierto tamaño de información.

Para el proyecto se decidió ir por la versión en la nube. Si bien localmente el espacio no es una restricción (limitado al tamaño del disco), para la distribución de

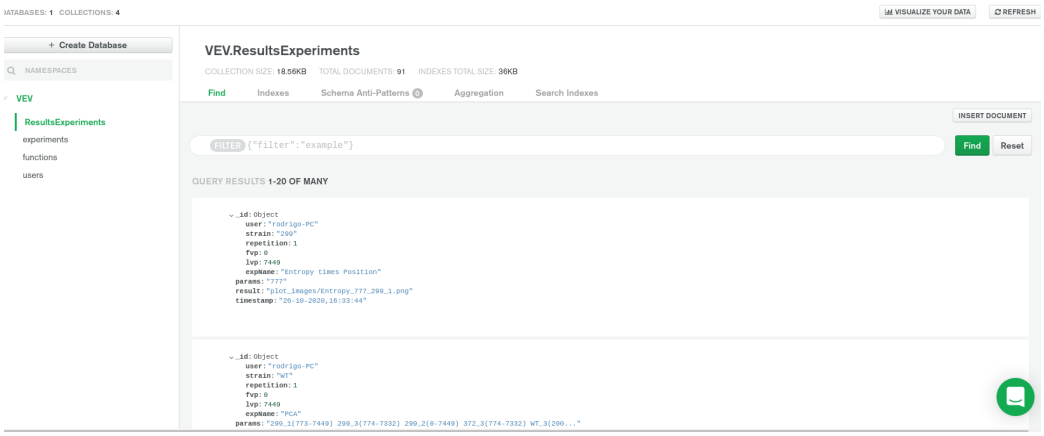


Figura 6.19: Ejemplo en Atlas MongoDB para la colección de resultados de análisis de experimentos

la herramienta sería necesario que el usuario tenga instalado MongoDB localmente y que además modifique un archivo de configuración de forma de poder vincular la base de datos a la herramienta. Además, los resultados de los usuarios quedan almacenados en un repositorio el cual provee más seguridad de que los mismos no se pierdan, ya que la infraestructura de estos proveedores está preparada para recuperarse de desastres. Evaluando el costo beneficio se tomó la decisión de tener la base en la nube. Para esto se agregó la noción de usuario para identificar a quien pertenece un análisis en particular. Con este agregado, se implementó la funcionalidad que exporta la información de los resultados obtenidos por el usuario. Éste agregado llevó a que se tuviera que desarrollar una versión fuera de línea de la herramienta. Existen situaciones donde el usuario no tiene acceso a internet o por algún motivo no sea posible la comunicación con la base de datos. Para atender a esta problemática, se permite pasar como parámetro la palabra *offline* y de esta forma la herramienta no se conecta contra la base de datos. Al no conectarse a la misma todos los análisis ejecutados en esa sesión no se van a ver reflejados en ella.

En este proyecto toda entrada a la base de datos se ingresa automáticamente desde la herramienta, es decir, no hay interacción directa del usuario. Sin embargo, se utilizó JSON-SCHEMA de forma de validar las entradas a la base de datos ya que como se ha mencionado, MongoDB es de esquema libre. Utilizando el mismo, se puede comprobar la correctitud de lo ingresado y en un futuro, de ser necesario, validar las entradas del usuario.

JSON-SCHEMA tiene como uso principal describir la estructura y las restricciones de validación de los documentos JSON. Para el proyecto se utilizó la herramienta en línea **JSON Schema Tool** [27]. En ésta se escribe un ejemplo de un documento JSON válido para cada una de las colecciones que utiliza y la misma herramienta retorna el esquema para validar estas entradas. En el contexto del proyecto, se tienen cuatro colecciones distintas, y por ende, se generaron cuatro esquemas diferentes.

6.4. Distribución de la Herramienta

Para la distribución de la herramienta se crearon dos formas distintas de manera de poder simplificar al usuario y que éste adopte la opción que sea mas sencilla para él.

6.4.1. GitHub

Se creó un repositorio en Github [19], el cual es público con todo el proyecto desarrollado, sin agregar los datos de los experimentos. Se decidió utilizar Github ya que es muy utilizado en general y además es muy sencillo descargar un repositorio. Se creó un archivo con el instructivo de instalación, el cual contiene dos maneras de instalar las bibliotecas necesarias para el proyecto.

- Utilizando Anaconda [2] con Python
- Utilizando el comando pip [45] sin anaconda.

Dado que Anaconda es muy utilizado y las versiones de librearías así como la instalación es distinta, se tomó la decisión de también distribuir la herramienta de esta manera.

6.4.2. PyPI

Se agregó la herramienta para que pueda ser directamente instalable desde PyPI [42]. El Python Package Index (PyPI) es un repositorio de software para el lenguaje de programación Python. Como muchas herramientas disponibles en Python, éstas se pueden descargar fácilmente utilizando el comando pip y escribiendo el nombre de la herramienta. Esta herramienta se instala ejecutando: —pip3 install vev-fing—. Luego únicamente utilizando el comando `vev` desde linea de comandos la herramienta se ejecuta.

Capítulo 7

Conclusiones y trabajo futuro

En este capítulo se presentan las conclusiones del proyecto así como también las líneas de trabajo futuro.

7.1. Conclusiones

Se desarrolló exitosamente una herramienta de análisis y visualización de la evolución para virus ARN que aporta valor al usuario. Se estudiaron los métodos de análisis desarrollados en el proyecto anterior. Asimismo, se interpretaron los resultados con ellos obtenidos y se nuclearon en la herramienta. Para ello, se estudiaron e implementaron nuevas estructuras para el almacenamiento y procesamiento de los datos de manera de posibilitar trabajar con virus de ARN genérico.

Se agregan dos nuevos métodos de análisis y visualización. Uno, t-SNE más enfocado en distintas visualizaciones con las que el usuario puede trabajar de forma de detectar posibles trayectorias atípicas. El otro, agrupaciones de series temporales, el cual no solamente aporta una capa de visualización de cómo se agrupan las distintas posiciones en los distintos grupos, sino que también aporta información de las posiciones que en ellos se agrupan. Para los grupos más pequeños se despliega al usuario las distintas posiciones de posible interés para su posterior análisis. Se implementa una base de datos de forma de almacenar información relevante para el usuario de forma de poder retornarle la misma a demanda.

Se implementó una interfaz que busca ser amigable con el usuario, donde éste puede interactuar de manera sencilla con todos los métodos, pudiendo trabajar de forma paralela. Con la herramienta se entrega un manual de usuario.

Por todo esto se concluye que se cumplieron todos los objetivos establecidos en la sección 1.2

7.2. Trabajo futuro

A continuación se listan las distintas líneas de trabajo futuro así como posibles mejoras.

La herramienta fue diseñada inicialmente para ser ejecutada localmente en la estación de trabajo del usuario. Aunque se empezó a trabajar para obtener una arquitectura distribuida utilizando una base de datos donde se almacene información de interés, todo el procesamiento se ejecuta localmente, así como también es necesario tener la información de los experimentos de manera local. A futuro podría ser deseable trabajar en continuar desacoplando la herramienta de manera de poder tener la capa de procesamiento en servidores con alto poder de cómputo, así como también la información de los experimentos. De esta manera, el usuario únicamente tiene un cliente liviano donde interactuar con los métodos. Esto también facilitaría compartir la información de los experimentos, y evitar que cada usuario los almacene de manera local.

Otra línea de mejora sería poder proveer la posibilidad de trabajar no únicamente con el genoma del virus de forma total, si no poder aplicar los distintos métodos a las distintas regiones de importancia biológica que presenta de forma de poder analizarlos individualmente.

También se podría agregar a la visualización de t-SNE, la posibilidad de resaltar posiciones a demanda por el usuario, sin tener que ubicarla con el cursor del ratón. Si bien la información se encuentra en la gráfica, al momento de buscar una posición en particular, y en el caso de que existan muchas posiciones distintas, es relativamente complejo localizar una posición en particular.

Como se menciona en el capitulo 6, cuando se trabaja fuera de línea en la herramienta, los análisis efectuados no se ven reflejados en la base de datos. Como una mejora a futuro, sería interesante que esta información quede almacenada localmente y en la próxima sesión que sea en línea, se registre esta información en la base de datos.

Puede ser interesante, evaluar la usabilidad y sencillez de la herramienta con el fin de poder evaluar la interfaz de usuario desarrollada.

Por último, otro posible trabajo futuro, sería poder analizar y agregar al método de Clustering, la posibilidad de trabajar con series temporales multivariadas con las distintas componentes SSA y así poder analizar el comportamiento del método.

Apéndice A

Anexo Herramientas Utilizadas

Para el desarrollo de esta herramienta se utilizó Python 3 con sus librerías nativas, en su versión 3.6.9 y otras tales como: Pandas, scikit-learn, Numpy, Matplotlib, PyQT5, Scipy y Pillow.

A.1. PyQT

En el capitulo 6 en la sección de interfaz, se menciona que se utiliza PyQT5 como framework para trabajar en la interfaz gráfica. En esa sección ya se desarrollaron los motivos para elegir la misma. Éstas son:

1. Posibilidad de construir una interfaz con un diseño agradable para el usuario
2. Buena documentación
3. Posee Herramienta para facilitar el trabajo de diseño
4. Simpleza para adaptar el código

PyQt es una biblioteca que le permite usar el marco de interfaz gráfica de usuario (GUI) Qt en Python ya que Qt está escrito en C ++. Con PyQt5 se hace referencia a la versión 5 de PyQT la cual es la última versión disponible.

Posee grandes ventajas, algunas de las cuales se listan a continuación:

- La programación de interfaz de usuario con Qt está diseñada entorno al concepto de señales y ranuras para establecer la comunicación entre objetos. Permite flexibilidad cuando se trata de eventos de GUI.
- Ofrece varios widgets, como botones o menús, todos diseñados con una apariencia básica en todas las plataformas compatibles.
- PyQt es uno de los marcos de interfaz de usuario más utilizados en Python, por lo tanto existe amplia documentación y ejemplos.
- Provee una herramienta propia para facilitar el diseño de la interfaz de usuario.

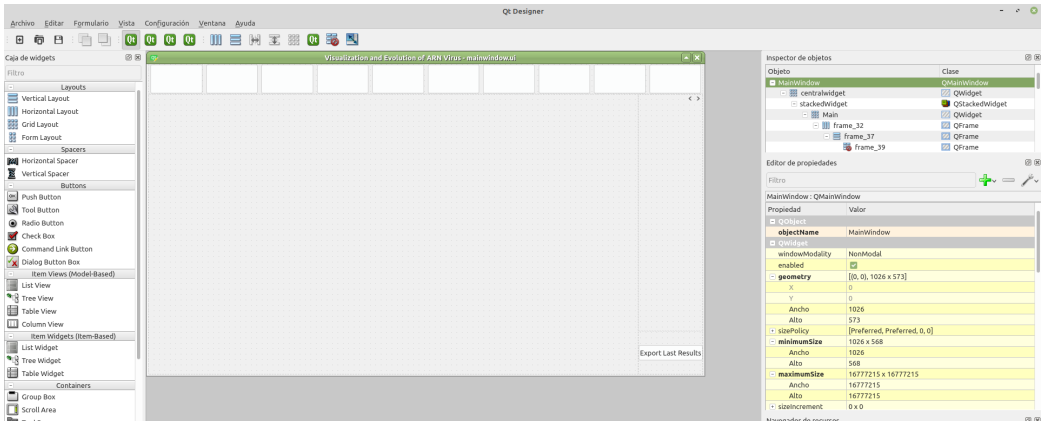


Figura A.1: VEV en QTDesigner

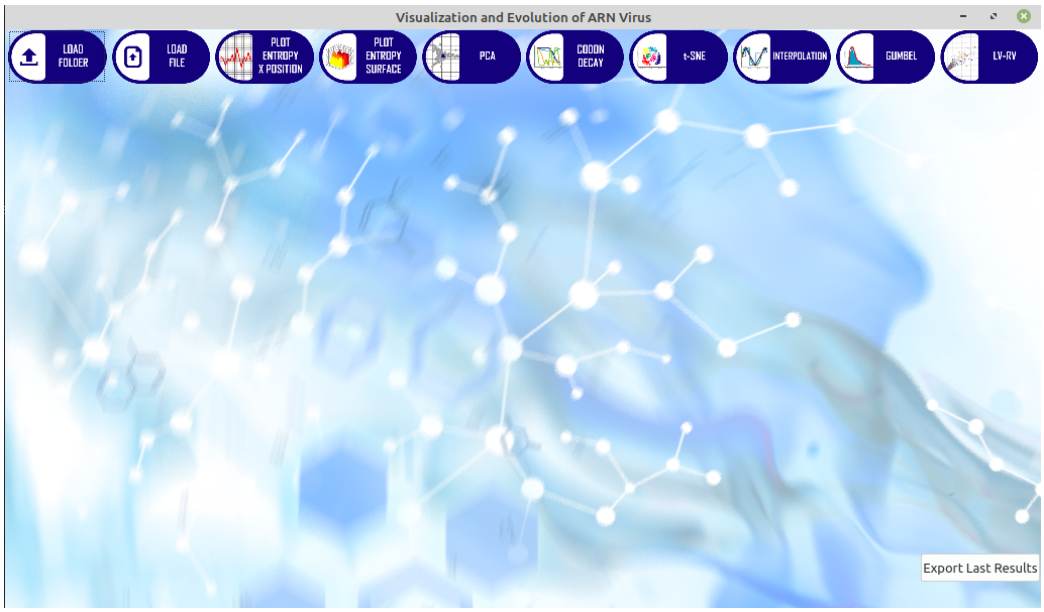


Figura A.2: VEV

A.1.1. QtDesigner

Al no tener conocimientos de diseño en el equipo, se buscó alguna herramienta que facilitara aunque sea un poco este trabajo. Con QtDesigner se puede diseñar toda la herramienta, ya sea las clases a utilizar, contenedores, botones, ubicación, etc. Presenta una paleta donde se encuentran todos los tipos de objetos que se pueden utilizar y simplemente arrastrando y soltando se logra agregar el objeto requerido al diseño. Si bien no se completa totalmente el diseño de la herramienta utilizándolo, sí simplifica, y al permitir exportar el archivo como en formato .UI el cual es un archivo XML, es fácilmente cargado desde el módulo Python. En las figuras A.1 y A.2 se puede observar como se ve la herramienta desde QTDesigner y con los estilos cargados.

A.2. Numpy

NumPy es una biblioteca de Python numérica de código abierto. Utiliza estructuras de datos matriciales. Como la biblioteca contiene una gran cantidad de funciones algebraicas y de transformaciones, se suele utilizar para realizar operaciones matemáticas en matrices, como rutinas trigonométricas, estadísticas y algebraicas [21]. Surge como una extensión de Numeric y Numarray [53].

Es la biblioteca central para la computación científica en Python. Esto no va en detrimento de Pandas, ya que ambas trabajan juntas, al punto de que se puede hablar que los objetos Pandas dependen en gran medida de los objetos NumPy. En otras palabras, esencialmente, Pandas extiende a Numpy [21].

El objeto principal de NumPy es una matriz multidimensional homogénea. Es una tabla de elementos (generalmente números), todos del mismo tipo, indexados por una tupla de enteros positivos. Esto tiene una parte positiva a nivel de procesamiento, pero como contrapartida, no se puede trabajar con objetos que tengan distintos tipos de datos en la misma columna. Para resolver esto se utiliza Pandas [53].

La versión de Numpy utilizada fue la versión 1.18.4 la cual fue publicada en Mayo del 2020.

A.3. Scikit-learn

Scikit-learn es una librería de Python que integra una amplia gama de algoritmos de aprendizaje automático de última generación para problemas supervisados (y no supervisados) de mediana escala. La idea de esta librería, se centra en llevar el aprendizaje automático a los no especialistas que utilizan un lenguaje de alto nivel de propósito general (en este caso Python). Se enfatiza la facilidad de uso, el rendimiento, la documentación y la coherencia de la API. Tiene dependencias mínimas y se distribuye bajo la licencia BSD, lo que fomenta su uso en entornos académicos y comerciales [40]. Las licencias BSD (Berkeley Software Distribution) son una familia de licencias de software libre permisivas, que imponen restricciones mínimas sobre el uso y la distribución de software cubierto. La licencia BSD es una licencia simple que solamente requiere que todo el código conserve el aviso de licencia BSD si se redistribuye en formato de código fuente, o reproduce el aviso si se redistribuye en formato binario [37].

Como se menciona, los puntos fuertes son, la consistencia y calidad del código, no intentan desarrollar tantas herramientas como puedan sino, proporcionar herramientas sólidas. Una API sencilla de utilizar. Mucha documentación y desarrollada por la comunidad [40].

Scikit-Learn, utiliza muchas de las librerías que se utilizan en el proyecto, por lo cual es muy sencillo adaptarlo. Tecnologías utilizadas por Scikit-Learn [40]:

- NumPy (de la cual se habló anteriormente) para la estructura de datos base, utilizada para datos y parámetros del modelo. Los datos de entrada se presentan como matrices NumPy.

- Scipy (la cual se hablará luego) cuenta con algoritmos eficientes para álgebra lineal, representación de matriz dispersa, funciones especiales y funciones estadísticas básicas.

El objeto central es un estimador, que implementa un método de ajuste, aceptando como argumentos una matriz de datos de entrada y, opcionalmente, una matriz de etiquetas para problemas supervisados. Los estimadores supervisados, como los clasificadores SVM, pueden implementar un método de predicción. Algunos estimadores, llamados transformadores (un ejemplo es el PCA, el cual se utiliza en el proyecto), implementan un método de transformación, devolviendo datos de entrada modificados [40]. Este último ejemplo se puede ver mejor en la sección 6.3.2.

La versión utilizada fue la 0.22.2.post1 la cual fue publicada en marzo de 2020.

A.4. Matplotlib

Matplotlib es una librería de Python (la cual no es necesario instalarla aparte) que sirve para graficar. Es una biblioteca muy poderosa útil para aquellos que trabajan con Python y NumPy. El módulo más utilizado de Matplotib es Pyplot, que proporciona una interfaz como MATLAB pero, en cambio, utiliza Python y es de código abierto [22].

Uno de los requerimientos mas importantes de este proyecto, era la parte de visualización. Para esto se necesitó utilizar alguna librería que permitiera graficar de manera sencilla, los distintos tipos de visualizaciones. Además debía permitir una fácil relación con las otras liberías utilizadas. Por último debía permitir la posibilidad de trabajar con eventos.

Matplotlib cumple con todos estos requerimientos, aunque no es la única.

Una gráfica en matplotlib consta de distintos objetos, los cuales tienen una relación jerárquica. Estos son [22]:

- Figura: es la figura completa. Puede contener varios Axes. La figura es la unidad que contiene todas las otras partes.
- Axes: Son la trama o gráfico individual.
- Axis: Son los ejes (ejemplo: x e y) y se encargan de generar los límites del gráfico.
- Arista: todo lo que se puede ver en la figura es una artista, como objetos de texto, Line2D, objetos de colección. La mayoría de las aristas están vinculados a los Ejes (Axes).

En la figura A.3 se puede ver esa jerarquía.

Se utilizó principalmente el módulo PyPlot. Pyplot es una colección de funciones que hacen que Matplotlib funcione como MATLAB (MATrix LABoratory, sistema de cómputo numérico que ofrece un entorno de desarrollo integrado y con

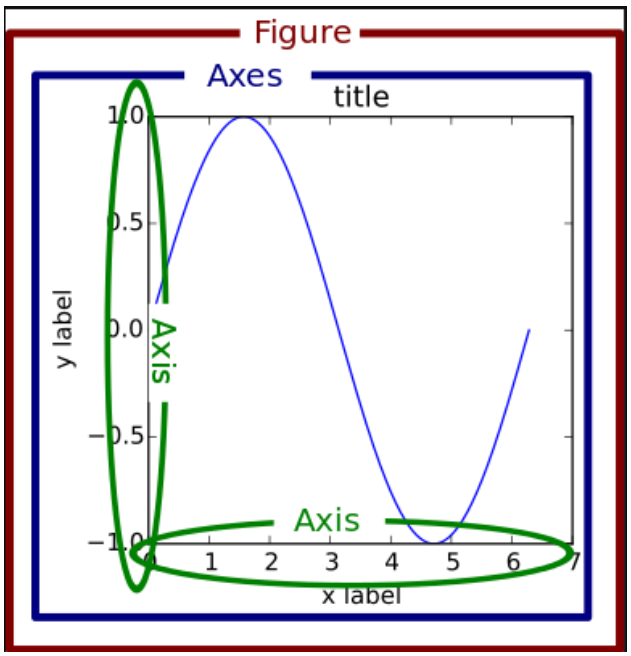


Figura A.3: Jerarquía Objetos Matplotlib [47]

un lenguaje de programación propio). Cada función de Pyplot hace algún cambio en una figura [34].

Por los requerimientos funcionales del proyecto, algunas de las figuras debían permitir el manejo de eventos. Matplotlib, funciona con una serie de kits de herramientas de interfaz de usuario (en particular y en este proyecto, trabaja con PyQT) para poder interactuar con la figura mediante pulsación de teclas o eventos de cursor. Para esto utiliza la función `mlp_connect`. Un ejemplo de evento podría ser cuando el usuario hace un click [35].

La versión de matplotlib que se utilizó en el proyecto es la 3.2.1 la cuál se publicó en abril de 2020

A.5. Scipy

SciPy es un conjunto de herramientas científicas y numéricas de código abierto (con licencia BSD) para Python. Proporciona distintas clases y comandos para la manipulación de datos [52].

Si bien Numpy provee funcionalidades matemáticas, para cálculos mas complejos, es necesario utilizar Scipy conjuntamente con Numpy. Esto es debido a que utiliza las estructuras de datos que ésta proporciona. A modo de ejemplo, si bien Numpy posee funciones álgebra, Scipy contiene más funcionalidades [52].

En el contexto del proyecto, se utilizó para resolver algunos cálculos tales como promedio móvil, aproximación de la distribución de Gumbel, así como también, probar ciertos resultados ya implementados como el cálculo de la entropía de Shan-

non.

A.6. Pillow

Pillow es una biblioteca de Python para la manipulación de imágenes. Pillow surge a partir de la biblioteca PIL. Ésta última es una biblioteca que ofrece varias funcionalidades para manipular imágenes. Si bien PIL es una biblioteca muy poderosa, no se ha actualizado desde 2011 y no es compatible con Python 3. Pillow agrega más funciones y soporte para Python 3. Es compatible con una variedad de formatos de archivos de imagen como PNG, JPEG, PPM , GIF, TIFF y BMP [6].

En éste proyecto, ésta biblioteca se utilizó para poder cumplir con algunos requerimientos que Matplotlib por si solo no podía solucionar.

La versión utilizada fue la 7.1.2 publicada en abril de 2020.

Siglas

ADN Ácido desoxirribonucleico.

API Application Programming Interface.

ARN Ácido ribonucleico.

BSD Berkeley Software Distribution.

CSV Ccomma Separated Values.

DTW Dynamic Time Warping.

FVP First Valid Positions.

GUI Graphical User Interface.

HDF Hierarchical Data Format.

HF High Fidelity.

JSON JavaScript Object Notation.

LF Low Fidelity.

LR Learning Rate.

LV Leading Variations.

LVP Last Valid Positions.

MB Megabyte.

MiB Mebibyte.

NGS Next Generation Sequencing.

PCA Principal Component Analysis.

Siglas

PyPI Python Package Index.

RLE Run-Length Encoding.

RV Random Variations.

SNE Stochastic Neighbor Embedding.

SSA Singular Spectrum Analysis.

SVD Singuar Value Descomposition.

t-SNE t-Distributed Stochastic Neighbor Embedding.

WT Wild Type.

XML Extensible Markup Language.

Glosario

Ácido nucleico Los dos tipos básicos de ácidos nucleicos son el ADN y el ARN. Están compuestos por una cadena de nucleótidos unidas mediante enlaces fosfodiéster.

Alelo Una de las diferentes formas de un gen que pueden darse en un único locus.

Aminoácido Elemento básico del que están formados las proteínas y péptidos.

ARN mensajero Molécula de ARN transcrita del ADN de un gen cuya traducción en los ribosomas genera una proteína.

Capsómero Subunidades morfológicas de la cápside. Cada capsómero puede estar constituido por una o varias proteínas.

Citoplasma Material comprendido entre las membranas celular y nuclear.

Codón Conjunto de tres nucleótidos adyacentes, no solapados que cifra un único aminoácido.

Cápside Es una estructura proteica que envuelve el ácido nucleico de un virus. Formada por una serie de monómeros llamados capsómeros.

Enlace Fosfodiéster Enlace covalente que se produce entre un grupo hidroxilo y un grupo fosfato.

Espícula Glicoproteína en forma de espícula presente en una cápside viral o una envoltura vírica. Son esenciales tanto para la especificidad de huésped como para la infectividad viral.

Evaginación Salida de un órgano hacia fuera de la vaina, saco o cavidad donde normalmente está contenido.

Fenotipo Manifestación externa observable de un genotipo.

Filtro pasabajos En señales en general, un filto pasabajos corresponde a aquel que devuelve una señal **suavizada**. Aquellas oscilaciones *bruscas* serán filtradas dejando pasar los cambios *suaves* o más lentos. Puesto que el ruido en una señal es una componente que oscila muy rápidamente, será filtrado con éste tipo de filtros.

Gen Unidad fundamental, física y funcional, de la herencia genética. Almacena la información genética y permite transmitirla a la descendencia.

Genotipo Composición alélica específica de una célula, bien referida al total de su genoma o, más comúnmente, a un gen determinado o a un conjunto de genes.

Glicoproteína Moléculas compuestas por una proteína unida a uno o varios glúcidos, simples o compuestos.

Glúcidos Moléculascompuestas principalmente de carbono, hidrógeno y oxígeno.

Grupo fosfato Molécula presente en el ARN y ADN formada por fosforo y oxigeno.

Locus Sitio preciso de un cromosoma donde está situado un gen.

Membrana plasmática Membrana plasmática o membrana celular, es una capa que delimita toda la célula, dividiendo el medio extracelular del intracelular.

Molécula heterocíclica Molécula cíclica (es decir que forma un anillo o lazo) que contienen átomos de más de un elemento en el ciclo.

Nucleótidos Molécula formada por la unión de un nucleósido y un grupo fosfato.

Oligonucleótido Secuencia corta de ADN o ARN, con cincuenta pares de bases o menos.

Patógeno Organismo causante de una enfermedad en otro organismo.

Pentámero En una cápside existen distintos vertices donde se encuentran los capsómeros. El capsómero en el vertice que tiene cinco vecinos se denomina pentámero.

Polimerasa Enzima que cataliza la síntesis de una cadena de ARN empleando un ADN como molde.

Protómero Unidad estructural de proteínas oligoméricas. Los capsómeros son proteínas individuales compuestas de protómeros.

Ribosoma Orgánulo complejo que cataliza la traducción del ARN mensaje en una secuencia de aminoácidos.

Transcriptasa Enzima que cataliza la síntesis de ADN utilizando un molde de ARN.

Virión Partícula vírica morfológicamente completa e infecciosa.

Viropexis Procesoque permite que algunos tipos de virus, tanto desnudos como con envoltura, se inserten en las células.

Referencias

[1] Khan Academy. https://es.khanacademy.org/science/high-school-biology/hs-molecular-genetics/hs-rna-and-protein-synthesis/a/hs-rna-and-protein-synthesis-review. Accessed on 2020-12-12.

[2] Anaconda. https://www.anaconda.com/. Accessed on 2020-12-12.

[3] Apache. Documentation. https://parquet.apache.org/documentation/latest/. Accessed on 2020-06-14.

[4] Apache. Hadoop. https://hadoop.apache.org/. Accessed on 2020-06-14.

[5] Juan Arbiza. http://www.higiene.edu.uy/cefa/2008/BiologiaViral.pdf. Accessed on 2020-12-12.

[6] Auth0. https://medium.com/@auth0/image-processing-in-python-with-pillow-af179f5a22e6. Accessed on 2020-06-14.

[7] David; Hendry-Reid; Luo Honglin; Yang Decheng; Ye Xin; Shi Junyan; McManus Bruce M B Garmaroudi, Farshid S; Marchant. Coxsackievirus b3 replication and pathogenesis. future microbiology, 10(4), 629–653.

[8] Jim Baggen, Hendrik Jan Thibaut, Jeroen Strating, and F. Kuppeveld. The life cycle of non-polio enteroviruses and how to target it. *Nature reviews. Microbiology*, 16, 04 2018.

[9] Snehotosh Banerjee. https://medium.com/@snehotosh.banerjee/feather-a-fast-on-disk-format-for-r-and-python-data-frames-de33d0516b03. Accessed on 2020-06-14.

[10] Antonio Barbadilla. La genÉtica de poblaciones. http://bioinformatica.uab.es/divulgacio/genpob.html. Accessed on 2020-06-13.

[11] Tarpey PS. Behjati S. What is next generation sequencing?. arch dis child educ pract ed. 2013;98(6):236-238.

[12] Martin Buncek. Griffiths, a.j.f., miller, j.h., suzuki, d.t., lewontin, r., gelbart, w.m.: An introduction to genetic analysis. *Biologia Plantarum - BIOL PLANT*, 45:50–50, 03 2002.

[13] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, USA, 2006.

[14] Michael W. Davidson and The Florida State University. `https://micro.magnet.fsu.edu/cells/virus.html`. Accessed on 2020-12-12.

[15] Feather. `https://arrow.apache.org/docs/python/feather.html`. Accessed on 2020-06-14.

[16] Feather. `https://github.com/wesm/feather/tree/master/python#limitations`. Accessed on 2020-06-14.

[17] Feather. Issues. `https://github.com/wesm/feather/issues/183`. Accessed on 2020-06-14.

[18] Genome. `https://www.genome.gov/es/genetics-glossary/Codigo-genetico`. Accessed on 2020-12-12.

[19] Github. `https://www.github.com/`. Accessed on 2020-12-12.

[20] Nina Golyandina and Anatoly Zhigljavsky. *Singular Spectrum Analysis for Time Series*. 01 2013.

[21] Killol Govani. `https://medium.com/fintechexplained/why-should-we-use-numpy-c14a4fb03ee9`. Accessed on 2020-06-11.

[22] Killol Govani. `https://towardsdatascience.com/matplotlib-tutorial-learn-basics-of-pythons-powerful-plotting-library-b5d1b8f67596`. Accessed on 2020-06-10.

[23] Guy Harrison. *Next Generation Databases: NoSQLand Big Data*. Apress, USA, 1st edition, 2015.

[24] HDF. `https://portal.hdfgroup.org/display/HDF5/Introduction+to+HDF5#IntroductiontoHDF5-hdf5desc`. Accessed on 2020-10-31.

[25] ICHI.PRO. `https://ichi.pro/es/t-sne-explicado-claramente-14863895622970`. Accessed on 2020-12-12.

[26] Anil Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31:651–666, 06 2010.

[27] JSON. `https://jsonschema.net/home`. Accessed on 2020-12-12.

[28] Kivy. `https://kivy.org/doc/stable/`. Accessed on 2020-06-05.

[29] Craig Larman. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition)*. Prentice Hall PTR, USA, 2004.

[30] Geoffrey Hinton Laurens van der Maaten. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9, 11 2008.

[31] Federico Lecumberry and Maria Ines Fariello. Proyecto csic i+d - análisis y visualización de la evolución de virus, 2017.

[32] E.P. Lessa. Guía de estudio de genética de poblaciones. *Laboratorio de Evolución de Facultad de Ciencias, Montevideo, Uruguay,2001.*

[33] Matplotlib. `https://matplotlib.org/`. Accessed on 2020-12-12.

[34] MatPlotLib. Event handling. `https://matplotlib.org/3.2.1/users/event_handling.html`. Accessed on 2020-06-12.

[35] MatPlotLib. Pyplot. `https://matplotlib.org/api/pyplot_api.html`. Accessed on 2020-06-12.

[36] Stephen Neidle. In Stephen Neidle, editor, *Principles of Nucleic Acid Structure*. Academic Press, New York, 2008.

[37] OpenSource. `https://opensource.org/licenses/bsd-license.php`. Accessed on 2020-06-12.

[38] Oracle. `https://docs.oracle.com/cd/E19528-01/820-0888/auto22/index.html`.

[39] Apache Parquet. `https://parquet.apache.org/`. Accessed on 2020-06-14.

[40] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, Edouard Duchesnay, and Gilles Louppe. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12, 01 2012.

[41] Theiss C Strutz-Seebohm N Seebohm G. Peischard S, Ho HT. A kidnapping story: How coxsackievirus b3 and its host cell interact.

[42] PyPI. `https://pypi.org/`. Accessed on 2020-12-12.

[43] PyQt. `https://www.riverbankcomputing.com/software/pyqt/`. Accessed on 2020-06-05.

[44] Python. Pickle. `https://docs.python.org/3/library/pickle.html#module-pickle`. Accessed on 2020-06-14.

[45] Python. Pip. `https://packaging.python.org/key_projects/#pip`. Accessed on 2020-12-12.

[46] Python. Tkinter. `docs.python.org/3/library/tkinter.html`. Accessed on 2020-06-05.

[47] Real Python. https://realpython.com/python-matplotlib-guide/. Accessed on 2020-06-12.

[48] QT. Qmainwindow. https://doc.qt.io/qtforpython/PySide2/QtWidgets/QMainWindow.html. Accessed on 2020-10-20.

[49] QT. Qtdesigner. https://doc.qt.io/qt-5/qtdesigner-manual.html. Accessed on 2020-06-05.

[50] Durrett R. Probability models for dna sequence evolution.

[51] Santiago Rubio, Rafael Adrián Pacheco-Orozco, Ana Milena Gómez, Sandra Perdomo, and Reggie García-Robles. Secuenciación de nueva generación (NGS) de ADN: presente y futuro en la práctica clínica. *Universitas Medica*, 61:49 – 63, 06 2020.

[52] SciPy. https://docs.scipy.org/doc/scipy/reference/tutorial/general.html. Accessed on 2020-06-13.

[53] SciPy. History of scipy. https://web.archive.org/web/20070927015018/http://www.scipy.org/History_of_SciPy/.

[54] Chathurangi Shyalika. https://medium.com/datadriveninvestor/dynamic-time-warping-dtw-d51d1a1e4afc. Accessed on 2020-12-12.

[55] Sushanth Sreenivasa. https://towardsdatascience.com/t-sne-behind-the-math-4d213b9ebab8. Accessed on 2020-12-12.

[56] StackOverflow. https://stackoverflow.com/questions/54665527/importerror-no-module-named-pandas-core-internals-managers-pandas-core-inte. Accessed on 2020-06-14.

[57] Romain Tavenard, Johann Faouzi, Gilles Vandewiele, Felix Divo, Guillaume Androz, Chester Holtz, Marie Payne, Roman Yurchak, Marc Rußwurm, Kushal Kolar, and Eli Woods. Tslearn, a machine learning toolkit for time series data. *Journal of Machine Learning Research*, 21(118):1–6, 2020.

[58] ViralZone. https://viralzone.expasy.org/33?outline=all_by_species. Accessed on 2020-12-12.

[59] WxPython. https://docs.wxpython.org/. Accessed on 2020-06-05.

[60] Ilia Zaitsev. https://towardsdatascience.com/the-best-format-to-save-pandas-data-414dca023e0d. Accessed on 2020-06-14.

[61] Jaime Zornoza. https://medium.com/swlh/the-mastery-of-pandas-i-50156db42125, 2019-10-19. Accessed on 2020-06-05.

Índice de tablas

2.1. Ejemplo de archivo de entrada. 14

6.1. Comparativa bibliotecas para Interfaz gráfica 51
6.2. Comparativa tiempos promedios en segundos para la carga y descarga de experimentos 69
6.3. Comparativa consumo en memoria RAM de experimentos en Mebibyte con distintas estructuras 69
6.4. Comparativa tamaño de experimentos en disco calculado en Megabyte con las distintas estructuras 70

Esta página ha sido intencionalmente dejada en blanco.

Índice de figuras

2.1. Ácidos nucleicos. A la izquierda de la figura ARN. A la derecha ADN [1]. 4
2.2. Código genético en el ARN [18]. 5
2.3. Ejemplo de formas geométricas de viriones [5]. 9
2.4. Esquema de replicación viral y sus etapas. Unión o *Attachment*. Penetración (en éste caso por viropexis o *endocytosis*). Liberación del material genético (*Uncoating*). Transcripción y replicación del genoma viral (*Translation* y *Genome replication*). Ensamblaje de las partículas hijas (*Assembly*). Maduración y liberación al medio extracelular (*Genome-induced virion maturation* y *Release*) [8]. . . 11
2.5. Virión de la familia Picornavirus a la que pertenece el Coxsackievirus [58]. 12
2.6. Modelo conceptual. 15

3.1. Reducción de la dimensión de los datos mediante entropía de Shannon de sesenta y cuatro a uno. 18
3.2. Simulación de evolución viral de los tres codones más frecuentes. Gráfica de sus frecuencias y la respectiva entropía. A la izquierda el escenario donde un codón es sustituido por otro. A la derecha el escenario donde un codón es sustituido por dos codones [31]. . . . 19
3.3. Mapeo de los valores de entropía a las regiones de la cápside. De izquierda a derecha, 372 (HF), WT, 299 (LF). Arriba pasaje 12, debajo el pasaje 20. En rojo las posiciones de entropía más alta en el genoma viral [31]. 20
3.4. Variación de frecuencia de codón dominante. Análisis de posiciones de interés mediante variación de frecuencia de codones dominantes. Umbral = 0,7. Cepa 299 repetición 3. 21
3.5. Evolución de la entropía en el tiempo(pasajes) para la posición 1995 de la cepa 299, repetición 1. 22
3.6. Descomposición LV-RV. Componentes LV y RV para el experimento de la cepa 299 repetición 1, posición 1995. 22
3.7. Análisis de espacio de variaciones para la cepa 299 con sus tres repeticiones. 23
3.8. Superficies de Entropía para la cepa 299 y repetición 1 y cepa WT y repetición 3, ambas con posiciones en el intervalo $[774, 7332]$. . . 24

3.9. Análisis de estadístico de los máximos de entropía para el experimento de cepa 299, repetición 2. 25
3.10. Descomposición PCA para las tres cepas con sus respectivos tres experimentos. Se grafica la trayectoria de la entropía para las posiciones cercanas a la representación de los experimentos (los nueve puntos de colores) mostrando como cercano a estos puntos, se encuentran posiciones con entropías altas correspondientes a estas cepas y en el centro, que son similares las entropías. 26

4.1. Conjunto de puntos en dos dimensiones mapeados a una dimensión [55]. 27
4.2. Gaussiana centrada en el punto de interés para cálculo de probabilidad condicional [25]. 29
4.3. Puntos de entrada para t-SNE por pasajes. 32
4.4. Visualización t-SNE por pasajes con valores *Perplexity*=6 y *Learning Rate*=100. Se observa que los datos son agrupados de acuerdo a su cepa y que en los primeros pasajes, es cuando los puntos de cada cepa más se asemejan. 33
4.5. Puntos de entrada para t-SNE por posiciones. 33
4.6. t-SNE por posiciones aplicado a la cepa 299 repetición 1. Se resaltan en rojo las posiciones de interés relevadas en el proyecto previo. . . 34

5.1. En la figura a se puede ver el camino DTW y en la figura b, la correspondencia en la serie temporal [54]. 37
5.2. SSA aplicado a una de las trayectorias de entropía. 39
5.3. Agrupación en dos y cuatro grupos de la repetición 1 de la cepa WT. 40
5.4. Agrupación en cuatro y seis grupos de la repetición 1 de la cepa WT. 41
5.5. Agrupación en seis y ocho grupos de la repetición 1 de la cepa WT. 41
5.6. Agrupación en ocho y diez grupos de la repetición 1 de la cepa WT. 41
5.7. Comparativa de agrupaciones para dos y cuatro grupos utilizando las trayectorias reales y la descomposición SSA. 43
5.8. Comparativa de agrupaciones para cuatro y seis grupos utilizando las trayectorias reales y la descomposición SSA. 44
5.9. Comparativa de agrupaciones para seis y ocho grupos utilizando las trayectorias reales y la descomposición SSA. 45
5.10. Comparativa de agrupaciones para ocho y diez grupos utilizando las trayectorias reales y la descomposición SSA. 46

6.1. Ejemplo inicio de hilo para el calculo de superficie de entropía . . . 52
6.2. PlotSurface hereda de QRunnable, la función *run* inicia al momento de la ejecución la función *start* desde el hilo principal 52
6.3. DataFrame Cepa 299, repetición 1, FVP 774, LVP 7332, con entropía 53
6.4. Diagrama de flujo de ejecución de funciones por el usuario 54
6.5. Directorio con Archivos CSV 55
6.6. Diálogo para la carga de experimentos ya procesados 56
6.7. Entropía por Posición 57

6.8. Superficie Entropía para cepa 299 repetición 1 57
6.9. Análisis de máximos de entropía para experimento con cepa 299, repetición 1, FVP 774 y LVP 7732. Se puede visualizar, por debajo de la gráfica las diez posiciones con menor $p - valor$ 58
6.10. Interpolación de entropía para experimento con cepa 299,repetición 1, FVP 774, LVP 7332 en pasaje 29 59
6.11. Análisis mediante variación de frecuencia para cepa 299, repetición 3, FVP 774, LVP 7332 con umbral 0,6 60
6.12. Descomposición PCA para tres experimentos con cepas distintas y trayectoria de entropía para posición 3558 con Entropía por posiciones 61
6.13. t-SNE aplicado a los pasajes, utilizando cinco experimentos. 62
6.14. t-SNE aplicado a pasajes, con parámetros LR=10 y Perplexity=3, filtrando la información de los puntos de la cepa WT 63
6.15. Aplicación de LV-RV por ambas aproximaciones, utilizando como parámetros cinco para la ventana deslizante y tres para el grado del polinomio 64
6.16. Aplicación del método Clustering con valores reales, para cepa 299, repetición 1, FVP 774, LVP 7332 utilizando seis grupos 65
6.17. Comparativa Tiempos Formatos 69
6.18. Ejemplo de Registro de actividad diario 70
6.19. Ejemplo en Atlas MongoDB para la colección de resultados de análisis de experimentos 71

A.1. VEV en QTDesigner 76
A.2. VEV 76
A.3. Jerarquía Objetos Matplotlib [47] 79

Esta es la última página.
Compilado el jueves 25 febrero, 2021.
https://www.fing.edu.uy/