UNIVERSIDAD CENTRAL "MARTA ABREU" DE LAS VILLAS

FACULTAD DE MATEMÁTICA, FÍSICA Y COMPUTACIÓN

APLICACIÓN DE LA SIMULACIÓN MONTE CARLO EN LA EVOLUCIÓN DE POBLACIONES DEL VIRUS DE INFLUENZA A/H1N1

Tesis presentada en opción al título académico de Máster en Ciencias de la Computación

Autor: TU NGUYEN THI MINH

Tutor: Dr. Robersy Sánchez Rodríguez

Dr. Ricardo Grau Ábalo

Santa Clara, 2011

DICTAMEN

Hago constar que el presente trabajo fue realizado en la Universidad Central "Martha Abreu" de Las Villas como parte de la culminación de los estudios de la especialidad de Ciencias de la Computación, autorizando a que el mismo sea utilizado por la institución, para los fines que estime conveniente, tanto de forma parcial como total y que además no podrá ser presentado en eventos ni publicado sin la autorización de la Universidad.

_____
Firma del autor

Los abajo firmantes, certificamos que el presente trabajo ha sido realizado según acuerdos de la dirección de nuestro centro y el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

_____
Firma del tutor

_____
Firma del Jefe del Seminario

PENSAMIENTO

"La completitud de Ciencia no es más que el refinamiento del pensamiento cada día"

__ Albert Einstein__

DEDICATORIA

A mi familia, por todo su amor que siempre me hace más fuerte.

A mi novio, por ser simplemente el hombre que siempre me calienta el corazón y el alma.

AGRADECIMIENTOS

Deseo agradecer a todas aquellas personas que de una forma u otra contribuyeron a que esta tesis fuera llevada a buen término.

De forma muy especial les doy las gracias a mis tutores, a Moreira, y a los demás miembros del Grupo de Bioinformática por su invaluable apoyo cada vez que lo necesité.

Muchas gracias a mi familia por su amor y el apoyo incondicional en todo momento.

Sinceramente agradezco también a mi amiga Mayra Morejón, por dedicarme siempre su amor, por cuidarme y animarme a pasar los momentos más difíciles de la vida.

Por último, pero no menos importantes, quiero agradecer a mis amigos, tanto vietnamitas como cubanos, porque con solo su presencia me dieron los ánimos necesarios para continuar trabajando.

RESUMEN

En este trabajo se exponen los resultados obtenidos en la simulación de la evolución de poblaciones del virus de la influenza A/H1N1, mediante la aplicación del método estocástico Monte Carlo y diferentes modelos evolutivos de Markov. El empleo de los modelos evolutivos y la estimación de sus parámetros, a partir de un conjunto de secuencias alineadas del gen de la Hemaglutinina, permite la aplicación de la simulación de Monte Carlo en la predicción de posibles variantes mutacionales. Con el propósito de evaluar las tendencias evolutivas de las poblaciones virales, en ausencia o presencia de la presión selectiva del sistema inmune, fueron implementadas dos variantes del algoritmo de generación de secuencias mutantes, utilizando el lenguaje de programación C++. Se aplica, además, un método de diseño experimental de superficies de respuesta para estimar los valores óptimos de los parámetros de la simulación, que maximizan la similitud entre las variantes mutacionales generadas y las observadas en la base de datos Influenza Virus Resources. Los parámetros utilizados fueron: el tiempo evolutivo, el tipo de procedimiento de simulación y el modelo evolutivo de Markov. Los resultados logrados sugieren el empleo de la simulación de Monte Carlo en la evaluación de las tendencias evolutivas de las poblaciones virales a corto plazo.

ABSTRACT

This paper presents the results obtained from the simulation of the evolution of influenza A/H1N1 virus populations, by means of the application of stochastic method Monte Carlo and different evolutionary models of Markov. The use of the evolutionary models and estimating its parameters, from a set of aligned sequences of the hemagglutinin gene, allow the application of Monte Carlo simulation in predicting possible mutational variants of the gen. In order to assess the evolutionary trends of viral populations, in the absence or presence of selective pressure of the immune system, were implemented two variants of mutant sequences generation algorithm by using the programming language C++. It was applied also an experiment design method of response surfaces to estimate the optimal values of simulation's parameters, that maximize the similarity between the mutational variants generated and the present in the Influenza Virus Database Resources. The parameters used were: the evolutionary time, the type of simulation procedure, and the markovian evolutionary model. The results obtained suggest the use of Monte Carlo simulation in the assessment of evolutionary trends of virus' populations to short term.

TABLA DE CONTENIDOS

INTRODUCCIÓN

Una de las áreas de la Biología que ha experimentado mayor expansión en los últimos años es la investigación de procesos evolutivos mediante la aplicación de técnicas de la Biología Molecular combinadas con los métodos matemáticos. Los resultados obtenidos en las investigaciones científicas han demostrado que la simulación es un método muy útil para validar una teoría o una implementación de un programa cuando el análisis del problema es complejo. Cuando el modelo es analíticamente intratable la simulación provee una manera potente para estudiarlo. La simulación de Monte Carlo, y en particular, el método de Monte Carlo Cadena de Markov (Markov-Chain Monte Carlo, MCMC) es el fundamento para formar una familia de algoritmos de generación de muestras aleatorias en un tiempo continuo y razonable. De hecho, debido a que el proceso de evolución molecular puede ser tratado como un proceso estocástico, la aplicación de MCMC resulta apropiada para resolver el problema de simulación de la evolución molecular de poblaciones de virus patógenos, un fenómeno confrontado diariamente por el sistema inmune del hombre. En particular, en el área de la medicina, es de vital importancia la introducción de tratamientos de los pacientes con nuevos antivirales, contra virus tales como el de la influenza A/H1N1 y el VIH, que cuales poseen una tasa de mutabilidad muy alta. Hasta el presente la comprensión de los procesos de evolución molecular es uno de los retos más encumbrados de la biología molecular evolutiva.

Considerando los antecedentes anteriores, se propone la idea de investigar el proceso de evolución de las poblaciones virales de la influenza A mediante la simulación de Monte Carlo. Para este fin se propone el uso de un modelo estocástico evolutivo que permita simular matemáticamente la aparición de nuevas variantes mutacionales en las poblaciones virales a partir de la introducción aleatoria de mutaciones en las secuencias tomadas como ancestros.

Por otra parte, la investigación de procesos evolutivos lleva aparejado el contraste de hipótesis alternativas. Existen modelos evolutivos que, bajo supuestos más o menos restrictivos, han permitido la construcción de pruebas paramétricas de contraste de hipótesis. Una de las dificultades que se encuentra en el desarrollo de estas pruebas es la derivación de las propiedades estadísticas correspondientes. En tales situaciones, es muy frecuente recurrir a la simulación de procesos de evolución para contrastar la bondad de esa derivación estadística. En este trabajo se realiza, además, un diseño experimental para estimar los valores óptimos de

los parámetros de entrada del algoritmo de simulación, en ausencia o presencia de la presión selectiva del sistema inmune.

Antecedentes y actualidad del tema

Los algoritmos de simulación, han sido ampliamente empleados en el análisis evolutivo de las secuencias genómicas de ADN en el campo de la Bioinformática. Pueden citarse algunos de los trabajos más significativos (Nielsen 2005; Beerli 2006; Kimmel 2006; Shao-Min Yan 2009) dedicados al desarrollo de métodos estadísticos en la evolución molecular; incluso, han sido desarrollados varios productos de software destinados enteramente a esta temática, algunos de los cuales pueden mencionarse como *HyPhy*, cuyo sistema está disponible en el sitio web www.hyphy.org, y *Evolver* (Robert C. E. 2009) que se encuentra en la página web www.drive5.com/evolver/.

Planteamiento del problema

Hasta el presente, en la literatura disponible, no hemos encontrado ninguna implementación de algoritmos que permitan simular la evolución de poblaciones virales, basado en los modelos evolutivos de Markov propuestos en el campo de la Biología molecular evolutiva. De hecho, podemos plantear la siguiente pregunta de investigación:

¿En qué medida la simulación de Monte Carlo de la evolución de poblaciones virales, usando los modelos evolutivos de Markov, permite predecir nuevas variantes mutacionales capaces de evadir la presión selectiva del sistema inmune?

Hipótesis del trabajo

Después de haber realizado un análisis del estado actual de la temática abordada, teniendo en cuenta los elementos teóricos antes presentados, así como las interrogantes existentes, se plantea la siguiente hipótesis general de investigación:

La simulación de Monte Carlo de poblaciones virales, utilizando los modelos evolutivos markovianos, es un procedimiento factible en la predicción in silico *de variantes mutacionales de escape, que evaden la acción del sistema inmune, inducida por la vacunación.*

Objetivos del trabajo

Como objetivo general de este trabajo se propone:

Construir un simulador integrado de la evolución de poblaciones virales, y aplicarlos particularmente al virus de la Influenza A/H1N1, mediante la técnica de simulación de Monte Carlo y los modelos evolutivos de Markov, el cual permite predecir nuevas variantes mutacionales capaces de evadir la presión selectiva del sistema inmune.

Con el propósito de cumplimentar este objetivo general, se proponen los siguientes objetivos específicos:

1. Implementar un algoritmo que permite simular la evolución de las poblaciones virales, en ausencia o presencia de la presión selectiva.

2. Implementar diferentes procedimientos de simulación y aplicarlos sobre los modelos evolutivos de Markov.

3. Integrar el software con los algoritmos de estimación de parámetros de los modelos markovianos.

4. Realizar los diseños experimentales, mediante el método de superficie respuesta, para estimar valores de los parámetros de simulación, que permiten descubrir variantes mutacionales con alto por ciento de similitud comparando con las secuencias reales encontradas en la base de datos *Influenza Virus Resources* disponible.

Contribución del trabajo

Este trabajo crea una nueva perspectiva en la aplicación de la simulación de secuencias genómicas en Bioinformática. En particular, la construcción de un sistema virtual de simulador de secuencias mutantes del virus, empleando diferentes modelos estocásticos evolutivos, para predecir nuevas variantes mutacionales que evaden la presión selectiva del sistema inmune, inducida por la vacunación de las poblaciones humanas, resulta novedosa, ya que, hasta el presente, no se han encontrado antecedentes en la literatura disponible. Por estas razones, este trabajo extiende los resultados previamente obtenidos en el grupo de Bioinformática y sienta la base importante para nuevos desarrollos de investigación en dicho grupo.

Estructura del trabajo

El trabajo se estructura esencialmente en tres capítulos.

- ✓ El Capítulo 1 está dedicado a la descripción de los fundamentos biológicos, matemáticos, y la introducción de las herramientas computacionales utilizadas, permitiéndole al lector tener un conocimiento general del tema abordado y una mejor comprensión de los capítulos siguientes.
- ✓ El Capítulo 2 se propone a alcanzar los 3 primeros objetivos del trabajo describiendo los algoritmos de simulación que permiten generar aleatoriamente las secuencias mutantes de ADN, bajo o sin condición selectiva, y aplicándolos sobre diferentes modelos evolutivos de Markov.
- ✓ El Capítulo 3 se dedica a diseñar los experimentos computacionales de simulación para determinar los valores de las variables dependientes incluidas en la simulación, a fin de optimizar la variable objetivo. De esta forma, se logra cumplir el último objetivo del trabajo.

Finalmente aparecen las conclusiones, recomendaciones, referencias bibliográficas y algunos anexos.

1 FUNDAMENTOS BIOLÓGICOS, MATEMÁTICOS Y COMPUTACIONALES

En este capítulo se realiza una descripción de las bases teóricas que conducen las aplicaciones de la simulación del proceso evolutivo de las poblaciones virales. Dado el hecho de que las aplicaciones son realizadas en secuencias de ADN genómico y que además, a lo largo de este trabajo, se aplica una variedad de herramientas matemáticas y computacionales, este capítulo se divide en cuatro secciones que permiten introducir al lector en la temática abordada.

1.1 Introducción a los términos biológicos

Se presentan en esta sección algunas ideas básicas de biología molecular y algunos resultados anteriores relacionados con este trabajo. Ellos son imprescindibles para la comprensión de las hipótesis de investigación y los capítulos siguientes.

1.1.1 Las moléculas básicas de la vida

Los principales actores de la Biología Molecular son las moléculas del ADN, ARN y las proteínas. Estas moléculas son simultáneamente los principales actores de un poderoso sistema de comunicación, el sistema de información genética, entrelazados por el código de comunicación más maravilloso conocido por el hombre, el Código Genético.

La expresión de la información almacenada en el Ácido desoxirribonucleico (ADN) se produce a través de la trascripción lineal de la secuencia de nucleótidos en la secuencia de nucleótidos del Ácido ribonucleico mensajero (ARNm). La secuencia de nucleótidos del ARNm es seguidamente traducida en una secuencia lineal de aminoácidos, de manera que el flujo de información es ADN → ARNm → Proteína.

1.1.2 El ADN y ARNm

El ADN está compuesto por cuatro moléculas básicas llamadas nucleótidos las cuales solo se diferencian en que cada una posee una base nitrogenada diferente. Cada nucleótido contiene un fosfato, un azúcar (desoxirribosa) y una de las cuatro bases nitrogenadas: *Adenina* (*A*), *Guanina* (*G*), *Timina* (*T*), *Citosina* (*C*) (*Figura 1.1*).

La estructura del ADN es una doble hélice. Las bases nitrogenadas se aparean en la doble hélice formando enlaces por puentes de hidrógeno de acuerdo a las siguientes reglas: *G≡ C, A = T*, donde cada "–" simboliza un enlace por puente de hidrógeno y cada par contiene una purina (*A* o *G*) y una piridamina (*C* o *T*). Las bases que forman el par se denominan bases complementarias. Una hélice simple es una cadena de nucleótidos unidos por enlaces fosfodiester. Cada hélice simple se une con la hélice complementaria a través de los enlaces de puentes de hidrógenos que forman las bases, dando lugar a la doble hélice (*Figura1.1*)

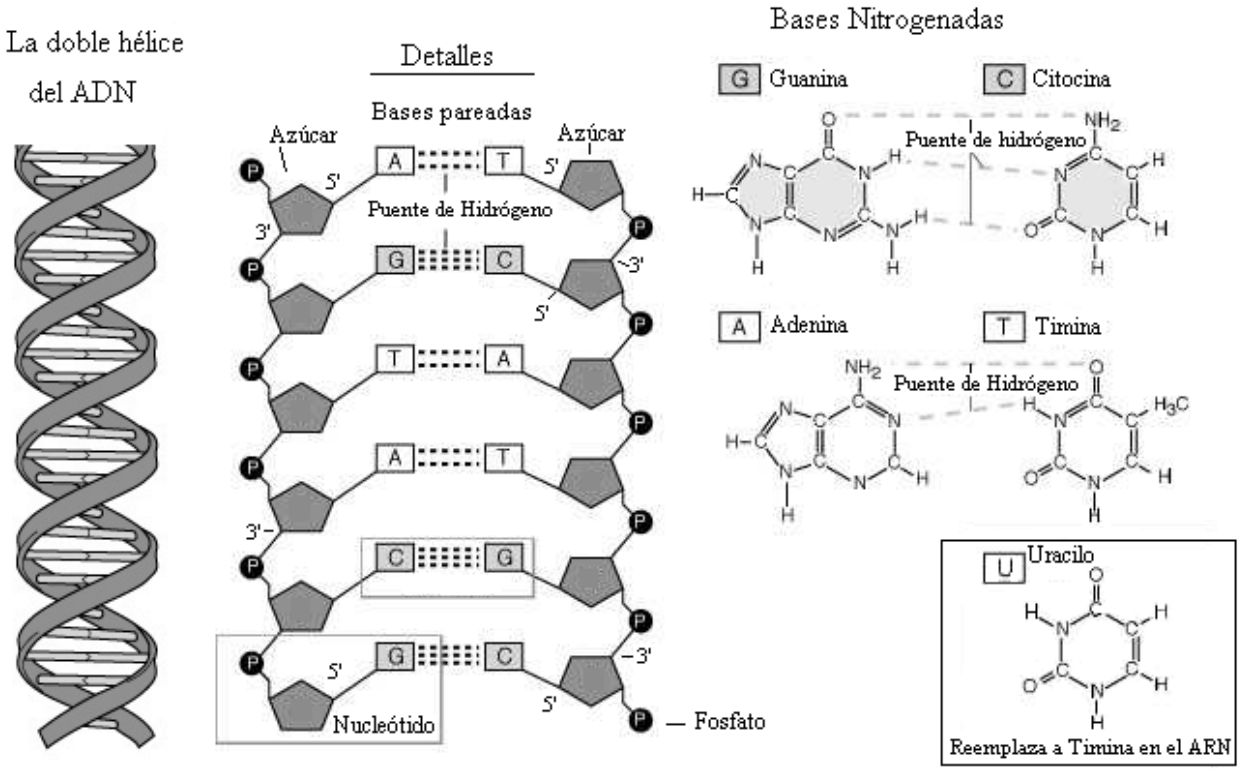


Figura 1.1. *La doble hélice del ADN. Detalles de las bases nitrogenadas y su apareamiento en el ADN.*

La molécula del ADN es una molécula direccional debido a la estructura asimétrica del azúcar desoxirribosa que forma el esqueleto de las hélices. Cada azúcar es conectado cadena arriba por el quinto carbono y cadena abajo por el tercer carbono, de manera que sí una de las cadenas tiene orientada el azúcar en la dirección 5'→3' (se lee de cinco prima a tres prima), la cadena complementaria estará orientada en la dirección 3'→5'. El ARNm está formado por

una cadena de nucleótidos unidos por enlaces fosfodiester. En particular constituye una hélice en la cual el azúcar presente en el nucleótido es una ribosa, la base nitrogenada T es sustituida por la base Uracilo (U) y se mantienen las otras bases nucleotídicas: A, C y G como en el ADN. La función del ARNm es transportar hacia el citoplasma la información genética que se encuentra almacenada en el núcleo de la célula en la secuencia de nucleótidos del ADN.
Cuando se traten secuencias de ADN codificantes para proteínas, éstas se pueden escribir en la práctica utilizando las bases T o U indistintamente.

1.1.3 Los aminoácidos y el código genético

Los aminoácidos son los sillares estructurales de las proteínas. Solo veinte aminoácidos se encuentran corrientemente presentes en las proteínas, aunque muchos otros aminoácidos desempeñan múltiples funciones en las células. La codificación de cada aminoácido se realiza a través de un grupo de tres nucleótidos. Si combinamos tres bases (tripletes) para formar un aminoácido, obtenemos un total de 64 combinaciones (4^3=64).

Los científicos demostraron que hay 61 tripletes -o codones- que codifican para aminoácidos, pero muchos de éstos son codificados por más de un codón, por lo que se dice que el código está degenerado. Los distintos aminoácidos son codificados por un número diferente de codones (algunos por 1, otros por 2, por 3, 4 o 6), e incluso existen tres tripletes que no codifican para ningún aminoácido. Encontramos que los aminoácidos como *Leucina*, *Serina* y *Arginina*, cada uno de los cuales posee 6 codones que codifican para el mismo, mientras que otros como el *Triptófano* y la *Metionina* solo poseen un solo codón (*Tabla 1.1*). El codón de los aminoácidos *Metionina* es usualmente utilizado en la mayoría de los seres vivos como codón de iniciación de la cadena polipeptídica. En resumen: de los 64 codones, 61 codifican aminoácidos y los tres restantes no son codificantes sino que son utilizados como señales de terminación y son llamados codones de parada (stop codons).

De la observación del código genético se destaca que la degeneración del código implica solamente a la tercera posición del codón en la mayoría de los casos (son excepciones la *Arginina*, la *Leucina* y la *Serina*) De esta forma resulta que las dos primeras bases de cada codón son las determinantes principales de su especificidad. La posición tercera, esto es, el

nucleótido situado en el extremo 3' del codón tiene menor importancia y no encaja con tanta precisión.

Tabla 1.1. *Tabla del Código genético estándar. Los aminoácidos están escritos con el símbolo de tres letras. El codón utilizado con mayor frecuencia como codón de inicio de la transcripción corresponde al aminoácido Metionina: AUG. Los codones UAA, UAG y UGA son marcadores del final de los genes.*

Primera base del Codón	Segunda base del Codón: U		Segunda base del Codón: C		Segunda base del Codón: A		Segunda base del Codón: G		Tercera base del Codón
U	UUU	Phe	UCU	Ser	UAU	Tyr	UGU	Cys	U
	UUC		UCC		UAC		UGC		C
	UUA	Leu	UCA		UAA	TER	UGA	TER	A
	UUG		UCG		UAG		UGG	Trp	G
C	CUU	Leu	CCU	Pro	CAU	His	CGU	Arg	U
	CUC		CCC		CAC		CGC		C
	CUA		CCA		CAA	Gln	CGA		A
	CUG		CCG		CAG		CGG		G
A	AUU	Ile	ACU	Thr	AAU	Asn	AGU	Ser	U
	AUC		ACC		AAC		AGC		C
	AUA		ACA		AAA	Lys	AGA	Arg	A
	AUG	Met	ACG		AAG		AGG		G
G	GUU	Val	GCU	Ala	GAU	Asp	GGU	Gly	U
	GUC		GCC		GAC		GGC		C
	GUA		GCA		GAA	Glu	GGA		A
	GUG		GCG		GAG		GGG		G

Existen muchas formas de clasificar los aminoácidos; las tres formas que se presentan a continuación son las más comunes:

1. Según las propiedades de su cadena:

Los aminoácidos se clasifican habitualmente según las propiedades de su cadena lateral:

- Neutros polares, polares o hidrófilos : Serina (Ser, S), Treonina (Thr, T), Cisteína (Cys, C), Asparagina (Asn, N), Glutamina (Gln, Q) y Tirosina (Tyr, Y).
- Neutros no polares, apolares o hidrófobos: Glicina (Gly, G), Alanina (Ala, A), Valina (Val, V), Leucina (Leu, L), Isoleucina (Ile, I), Metionina (Met, M), Prolina (Pro, P), Fenilalanina (Phe, F) y Triptófano (Trp, W).

- Con carga negativa, o ácidos: Ácido aspártico (Asp, D) y Ácido glutámico (Glu, E).
- Con carga positiva, o básicos: Lisina (Lys, K), Arginina (Arg, R) e Histidina (His, H).
- Aromáticos: Fenilalanina (Phe, F), Tirosina (Tyr, Y) y Triptófano (Trp, W) (ya incluidos en los grupos neutros polares y neutros no polares).

2. Según su obtención:

A los aminoácidos que necesitan ser ingeridos por el cuerpo para obtenerlos se les llama esenciales; la carencia de estos aminoácidos en la dieta limita el desarrollo del organismo, ya que no es posible reponer las células de los tejidos que mueren o crear tejidos nuevos, en el caso del crecimiento. Para el ser humano, los aminoácidos esenciales son: Valina (Val), Leucina (Leu), Treonina (Thr), Lisina (Lys), Triptófano (Trp), Histidina (His), Fenilalanina (Phe), Isoleucina (Ile), Arginina (Arg) y la Metionina (Met).

A los aminoácidos que pueden ser sintetizados por el cuerpo se los conoce como no esenciales y son: Alanina (Ala), Prolina (Pro), Glicina (Gly), Serina (Ser), Cisteína (Cys), Asparagina (Asn), Glutamina (Gln), Tirosina (Tyr), Ácido aspártico (Asp), Ácido glutámico (Glu).

Estas clasificaciones varían según la especie. Se han aislado cepas de bacterias con requerimientos diferenciales de cada tipo de aminoácido.

1.2 El alineamiento múltiple de secuencias

El alineamiento múltiple de secuencias de ADN y proteínas es la piedra angular de la Bioinformática ya que es el punto de partida para los análisis clásicos de secuencias desarrollados hasta el momento. Alinear es una forma de representar y comparar dos o más secuencias o cadenas de ADN, ARN, o estructuras primarias proteicas para resaltar sus zonas de similitud, que podrían indicar relaciones funcionales o evolutivas entre los genes o las proteínas analizadas. Las secuencias alineadas se escriben con las letras (representando aminoácidos o nucleótidos) en filas de una matriz en las que, si es necesario, se insertan espacios para que las zonas con idéntica o similar estructura se alineen.

Los alineamientos múltiples de secuencias son útiles para identificar similitudes y diferencias entre secuencias, producir árboles filogenéticos, y desarrollar modelos de homología sobre

estructuras de proteínas (*Figura 1.2*). Sin embargo, la relevancia biológica de los alineamientos no siempre es clara. Se asume a menudo que los alineamientos reflejan un grado de cambio evolutivo entre secuencias que descienden de un ancestro común; pero es formalmente posible que la convergencia evolutiva pueda darse para producir similitudes aparentes entre proteínas que no estén evolutivamente relacionadas, pero que lleven a cabo funciones similares y tengan parecidas estructuras.

Las representaciones visuales del alineamiento ilustran la inserción de espacios (gaps) que pueden ser interpretados como mutaciones (un solo cambio de aminoácido o nucleótido) que aparecen como diferentes caracteres en una sola columna del alineamiento, y la inserción o eliminación (*deletion*) de mutaciones, llamadas mutaciones "*indels*" que aparecen como huecos en una o varias de las secuencias en la alineación.

En la actualidad se dispone de muchos productos de software que permiten realizar el alineamiento múltiple de secuencias. Además, el servicio de alineamiento de secuencias se ofrece gratuitamente en servidores en Internet, aunque la mayoría de los softwares más populares se pueden adquirir libremente. En particular, en este trabajo fueron utilizados los softwares MEGA 4 y BioEdit, los cuales se pueden obtener en los sitios webs: http://www.megasoftware.net/ y http://www.mbio.ncsu.edu/BioEdit/bioedit.html, respectivamente. En la *Figura 1.2* se ilustra una región del alineamiento de secuencias codificantes para proteínas de virus influenza A/H1NN1 obtenido con el MEGA 4 (TAMURA K 2007).

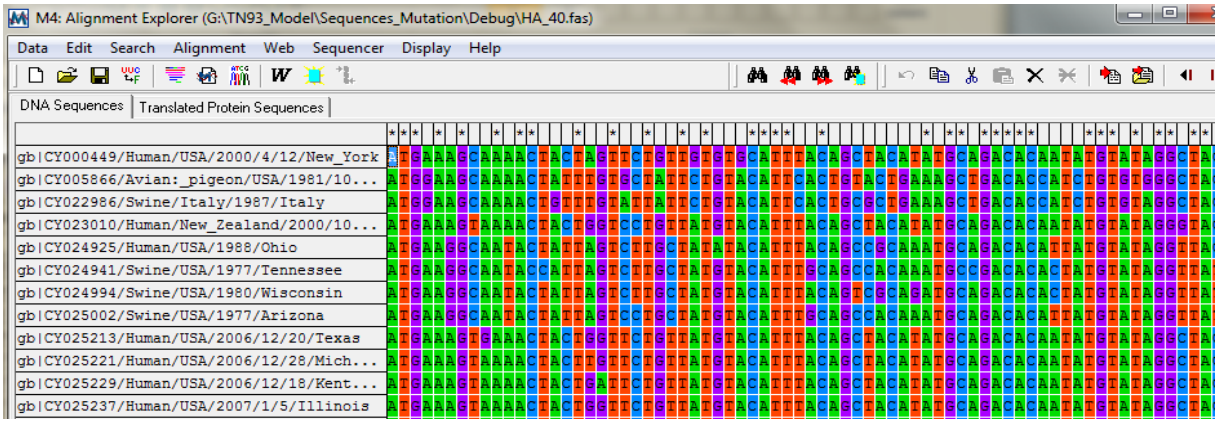


Figura 1.2. *Región de un alineamiento de secuencias codificantes para proteínas de virus influenza A/H1NN1*

1.3 La evolución molecular de las poblaciones

Cualquier organismo en la naturaleza vive en forma de poblaciones. Una población tiene propiedades que trascienden las características de un individuo. La genética de las poblaciones se dirige a la investigación de la estructura genética de las mismas y las reglas por las que la estructura genética se cambia. Al referirse a la estructura genética está hablándose de los tipos y las frecuencias de los genes y los genotipos presentados en la población (Nei, 1975).

La evolución es un proceso de transformaciones sucesivas de la estructura genética de las poblaciones. Los factores básicos de la evolución son: la mutación, la duplicación de genes, la selección natural y el cambio genético aleatorio (Nei, 1975). El objetivo del estudio de la evolución es comprender cuantitativamente todos los procesos evolutivos y ser capaz de predecir y controlar la evolución en el futuro de los organismos de las poblaciones.

El cambio evolutivo de los genes puede ser apreciado en considerable detalle estudiando el material genético ADN o sus productos directos de ARN y las proteínas de diferentes especies. Esto ha permitido a los genetistas evaluar los cambios evolutivos de las poblaciones de manera más cuantitativa y probar la validez de los supuestos previos sobre la evolución o la estabilidad de los sistemas genéticos.

Los cambios evolutivos de las secuencias de nucleótidos son muy lentos, ellos proveen informaciones detalladas de su origen y su historia. Ya que las secuencias de nucleótidos en los genes de las moléculas ADN son traducidas a las secuencias de aminos ácidos de las proteínas mediante el código genético, los cambios evolutivos de estas últimas también proveen información sobre el proceso y el tiempo aproximado de la evolución. De hecho, la mayoría de los resultados obtenidos a través de los estudios al nivel molecular viene de los análisis de las secuencias de aminos ácidos de ciertas proteínas.

1.3.1 La mutación

El estudio científico de la evolución empezó desde un artículo publicado por Darwin y Wallace en 1858. Primero, ellos postularon que la evolución ha ocurrido principalmente como resultado de la selección natural (Nei, 1975). Esta última es efectiva solo cuando existe la variación genética, y esta variedad genética es provista primeramente por la mutación.

Todas las características morfológicas y fisiológicas de los organismos son controlados por la información genética llevada en las moléculas ADN, las cuales son transmitidas de generación a generación. En algunos virus, la información genética es llevada en las moléculas ARN, pero la característica esencial de la herencia es la misma. En el proceso de desarrollo, la información genética contenida en la secuencia de nucleótidos ADN primero es transferida a la secuencia de nucleótidos de ARN mensajero (ARNm) por un proceso simple de transcripción uno a uno de los nucleótidos en la molécula ADN. Por el mismo proceso, se producen el ARN transportador (ARNt) y el ARN ribosomal (ARNr). La información transferida a ARNm determina después la secuencia de aminos ácidos de la proteína la cual será sintetizada. Los nucleótidos de ARNm son leídos secuencialmente 3 cada vez. Cada tal triplete o codón es traducido en un amino ácido en la cadena de la proteína mediante el código genético. Cualquiera de las mutaciones que son reconocidas como cambios morfológicos o fisiológicos tiene que ser debido a algún cambio de las moléculas ADN.

Existen cuatro tipos básicos de cambios en las moléculas ADN, ellos son: reemplazo de un nucleótido por el otro, eliminación de nucleótidos, adición de nucleótidos, e inversión de nucleótidos. La adición, la eliminación y la inversión pueden ocurrir con uno o más nucleótidos como una unidad. Los reemplazos de los nucleótidos pueden ser divididos en dos clases diferentes: la transición y el transversión. Transición es el reemplazo de una purina (adenin A o guanine G) por otro purine, o de un pyrimidine (thymine T o cytosine C) por otro pyrimidine. Otros tipos de sustitución de nucleótidos son llamados transversiones (Nei, 1975).

Los genes o segmentos de las moléculas ADN que actúan como las plantillas de las ARNm son llamados *genes estructurales.* Puesto que la secuencia de aminos ácidos es determinada por la secuencia de nucleótidos de un gen estructural, cualquier cambio en las secuencias de aminos ácidos es causado por la mutación ocurrida en la molécula ADN. Sin embargo, en el sentido opuesto un cambio de mutación en la molécula ADN no es necesariamente reflejado en el cambio de la secuencia de aminos ácidos. Esto es porque hay degeneración en el código genético (Nei, 1975).

1.4 Método de simulación de la evolución molecular

La simulación de Monte Carlo comúnmente ha sido utilizada en los estudios filogenéticos para probar diferentes métodos de reconstrucción de árboles filogenéticos y consecuentemente, sus aplicaciones para probar los modelos evolutivos pueden ser consideradas como una extensión natural de este uso. La simulación repetitiva de un proceso evolutivo dado, bajo las restricciones impuestas por el modelo que se prueba, permite estimar las distribuciones de probabilidad para los parámetros deseados. A continuación, el árbol filogenético se puede reconstruir de nuevo sin las restricciones del modelo, y los parámetros de interés derivados de este árbol pueden ser comparados con la distribución de probabilidad correspondiente derivada del árbol restringido y simulado (Yang, 2006).

1.4.1 Técnica de Markov Chain Monte Carlo

Los métodos de Markov Chain Monte Carlo, o simplemente métodos MCMC, a menudo se aplican para resolver los problemas de integración y optimización en los espacios de grandes dimensiones. Estos métodos pertenecen a una clase de métodos estocásticos y se usan cada vez más en la inferencia bayesiana y en la máquina de aprendizaje. En la evolución molecular, las primeras aplicaciones de MCMC han aparecido por la primera vez en los años 90, cuando varios autores intentaron a desarrollar métodos para calcular las probabilidades posteriores de filogenias sobre las secuencias de ADN alineadas (Nielsen 2005).

Uno de los objetivos básicos derivados de la teoría bayesiana general es calcular el valor esperado de una función *f* sobre un espacio paramétrico, con respecto a una distribución de probabilidad de alta dimensión $P(x_1, x_2,...,x_n)$, siendo x_i los valores de los parámetros o variables del modelo. La idea básica de MCMC consiste en, primero, estimar el valor esperado de *f* por:

$$E(f) = \sum_{x_1,x_2,...,x_n} f(x_1,x_2,...,x_n)P(x_1,x_2,...,x_n) \approx \frac{1}{T}\sum_{t=0}^{T} f(x_1^t,x_2^t,...,x_n^t), \qquad (1.1)$$

para cualquier $T>0$ y $(x_1^t,x_2^t,...,x_n^t)$ es generada como una muestra aleatoria, de acuerdo a su distribución $P(x_1,x_2,...,x_n)$. Segundo, la generación de las muestras se lleva a cabo mediante la construcción de una cadena de Markov, teniendo en cuenta *P* como la distribución estacionaria (Baldi and Brunak 2001).

Considerando un sistema $S = \{s_1, s_2, ..., s_n\}$ con n estados. Una cadena de Markov no es más que un conjunto $\{S^0, S^1, ..., S^t, ...\}$, donde cada elemento S^t representa un estado en un tiempo t determinado, es decir, en cualquier tiempo la cadena esta en un estado particular. Las variables S^t forman una cadena de Markov si y solo si para cualquier t, se tiene:

$$P(S^{t+1}|S^0, S^1, ..., S^t) = P(S^{t+1}|S^t) \quad (1.2)$$

Intuitivamente, esto puede expresarse de otra manera, diciendo que el futuro depende solamente al presente. Una cadena de Markov es totalmente definida por la distribución inicial $P(S^0)$ y las probabilidades de transición $P^t = P(S^{t+1}|S^t)$. Aquí nos referiremos sólo a las cadenas de Markov estacionarias, donde las probabilidades de transición son constantes, es decir, independientes del tiempo. Luego, la matriz de transición de la cadena es definida por $T=(t_{ij})$, donde t_{ij} es la probabilidad de transición del estado s_i al estado s_j.

Una manera para construir una cadena apropiada de Markov es usar un método conocido como una instancia de los métodos MCMC, llamado Metropolis. En este método, la idea de simulación de Monter Carlo es generar un conjunto de muestras dependientes, a partir de una distribución dada $P(S)$, las cuales se aceptan o se rechazan en dependencia de como la probabilidad del estado es afectada. Precisamente, el método Metropolis es definido usando dos familiares de distribuciones Q y R. De las cuales, $Q=(q_{ij})$ es la distribución de selección, donde q_{ij} es la probabilidad de seleccionar el nuevo estado s_j, dado el estado actual s_i. Mientras que $R=(r_{ij})$ es la distribución de aceptación, donde r_{ij} es la probabilidad de aceptar s_j como un posible estado siguiente, conociendo el estado actual s_i. Obviamente, se debe tener $q_{ij} \geq 0$, $r_{ij} \geq 0$ y $\sum_i q_{ij} = 1$. En la mayoría de los casos prácticos, se puede suponer que Q es simétrica, es decir $q_{ij} = q_{ji}$. Sin embargo, esta hipótesis luego fue modificada por Hastings, de forma tal que Q es asimétrica, por tanto el método Metropolis se convierte en un nuevo método llamado Metropolis-Hastings (Yang, 2006).

A partir del estado s_i en el tiempo t ($S^t = s_i$), el algoritmo de Metropolish procede de la siguiente forma:

1. Seleccionar aleatoriamente un nuevo estado s_j de acuerdo a la distribución q_{ij}.
2. Aceptar s_j con la probabilidad r_{ij}. Esto es, $S^{t+1} = s_j$ con la probabilidad r_{ij} y $S^{t+1} = s_i$ con la probabilidad 1- r_{ij}.

En la versión más común del algoritmo de Metropolis, la probabilidad de la aceptación se define por:

$$r_{ij} = \min\left(1, \frac{P(s_j)}{P(s_i)}\right) \tag{1.3}$$

Cuando se ignora la condición de que Q sea simétrica, es decir $q_{ij} \neq q_{ji}$, la probabilidad dada en (1.3) tiene la fórmula:

$$r_{ij} = \min\left\{1, \frac{P(s_j)}{P(s_i)} \cdot \frac{q_{ji}}{q_{ij}}\right\} \tag{1.4}$$

La distribución estacionaria P de una cadena de Markov y la probabilidad de transición $T(t_{ij})$ deben satisfacer la siguiente condición de balance:

$$P(S_i)\, t_{ji} = P(S_j)\, t_{ij} \qquad \forall i \neq j \tag{1.5}$$

En este caso, se dice que la distribución P es estable y las cadenas de Markov son reversibles, es decir, las tasas de transición de S_i a S_j es la misma que la de S_j a S_i (Nielsen 2005). Esta ecuación implica que una cadena de Markov corre hacia delante en el tiempo es indistinguible probabilísticamente de la misma cadena de Markov cuando esta corre hacia atrás. Comúnmente, la cadena de Markov se supone que es equilibrada, con un vector de distribución estacionario y es reversible en el tiempo. La reversibilidad es válida para un proceso en equilibrio cuando las ecuaciones de balance (1.5) mantengan válidas (Keyuan X. 2005).

En la sección siguiente veremos con más detalle de cómo se aplica el método de MCMC en la simulación de la evolución molecular y, en particular, el uso de los modelos markovianos para el proceso evolutivo de las secuencias genómicas.

1.4.2 Simulación del proceso evolutivo mediante el modelo de Markov

La caracterización de un modelo de Markov en los procesos evolutivos comienza con la definición de un alfabeto de residuos nucleotídicos. En el caso de las secuencias de ADN, el alfabeto consiste en el conjunto de 4 bases $S = \{A, C, G, T\}$.

La modelación del proceso de sustitución de nucleótidos en cualquier sitio particular de una secuencia de ADN, se describe por el método de cadena de Markov con los cuatro nucleótidos representando los estados de la secuencia. El aspecto principal de este método es que el mismo

no tiene memoria, es decir: dado el presente, el futuro no depende del pasado. En otras palabras, la probabilidad con la cual una posición de la cadena salta a los otros estados depende solo del estado actual y no importa cómo se ha alcanzado este último. Esto es conocido como "la propiedad markoviana" (Yang, 2006).

Todos los sitios de una secuencia evolucionan independientemente de acuerdo a la misma cadena de Markov. La simulación de la evolución en los sitios se manifiesta en el evento de la mutación, en la cual el lugar que ocurre una mutación y el nucleótido mutado es determinado aleatoriamente (E. Gultepe 2005). Un mecanismo que se usa para realizar este proceso aleatorio es la técnica de las cadenas de Markov descrita anteriormente, de forma que un nuevo nucleótido aleatorio generado s_j es aceptado como una sustitución del nucleótido original s_i ubicado en un sitio dado de la secuencia si se cumple que la probabilidad de (1.4) es mayor que un valor *v* aleatorio generado con distribución uniforme *U(0,1)*, de lo contrario se rechaza el nuevo nucleótido manteniendo el estado actual de la base que ocupa ese sitio. Este proceso se repite en toda la secuencia hasta que se genere una nueva que puede ser la mutación de la original o una idéntica a ella misma; en este último caso, no ha ocurrido ningún cambio en la evolución de la secuencia.

El modelo de evolución que utiliza un alfabeto extendido para codificar las moléculas ADN ha sido denominado por el Grupo de Bioinformática como "el modelo de las cinco bases" y lo abreviamos como modelo FB por sus siglas en inglés (Five Bases), o también, como es tradición en la denominación de los modelos evolutivos, como SG09 atendiendo a los apellidos de sus autores (Sánchez y Grau) y el año de su revelación (2009). En la literatura, existen varios modelos de Markov diferentes para la evolución de las secuencias de ADN. Esto es porque los procesos evolutivos cambian entre los genomas y entre las regiones de un genoma. Estos modelos difieren en la forma de la parametrización de la matriz de tasas de sustitución y en la modelación de la variación de las tasas (Sánchez and Grau, 2009). Como casos particulares del modelo FB, los modelos de cuatro bases difieren en que estos últimos no trabajan con el alfabeto extendido, es decir, la base hipotética D en el modelo FB no se presenta.

Cuando los modelos de Markov son continuos en el tiempo, se considera que la matriz $Q = \{q_{ij}\}$ representa la tasa de sustitución instantánea. Cada elemento q_{ij} de la matriz denota

un cambio del nucleótido *i* al *j, con i, j = A, C, G, T, y* las probabilidades de transición entre los estados vienen dadas por la relación *P(t)=exp(Qt)* que nos da la probabilidad de que cualquier nucleótido dado *i* se convierta al otro diferente *j* en un intervalo de tiempo pequeño $t > 0$. Esta matriz se conoce como *matriz de probabilidades de transición* (Yang 2006).

En los modelos de sustitución de nucleótidos, la forma estándar para calcular una función algebraica, como la exponencial, de una matriz *Q*, es diagonizarla. Suponiendo que *Q* pudiera escribirse en la forma:

$$Q = UDU^{-1}, \tag{1.6}$$

donde *U* es una matriz no singular de tamaño 4x4, que viene dada por el modelo de Markov, U^{-1} es su inversa, y *D* es una matriz diagonal creada a partir del vector de frecuencias de distribución de las bases de las secuencias ADN, $D = diag\{\lambda_1, \lambda_2, \lambda_3, \lambda_4\}$, el cual también se ha definido por el modelo de Markov. La ecuación (1.6) es conocida como la descomposición espectral de la matriz *Q*. De esta ecuación tenemos que:

$$Q^2 = (UDU^{-1}) \cdot (UDU^{-1}) = UD^2U^{-1} = U\, diag\{\lambda_1^2, \lambda_2^2, \lambda_3^2, \lambda_4^2\}\, U^{-1} \tag{1.7}$$

Similarmente $Q^m = U\, diag\{\lambda_1^m, \lambda_2^m, \lambda_3^m, \lambda_4^m\}\, U^{-1}$ para cualquier entero *m*. En general, cualquier función algebraica *f* aplicada a la matriz *Q* se puede calcular como

$$f(Q) = U\, diag\{f(\lambda_1), f(\lambda_2), f(\lambda_3), f(\lambda_4)\}U^{-1} \tag{1.8}$$

siempre y cuando *f (Q)* exista. Interesa aquí particularmente la función exponencial aplicada a la matriz *Qt* para construir *P*(*t*), que en esencia, es la solución de una ecuación diferencial matricial (realmente un Problema de Cauchy que se detalla posteriormente), y su existencia se garantiza con ciertas condiciones aquí presentes, gracias a la convergencia de una serie de potencias de la matriz *Qt*. Ver por ejemplo (Mederos O.; Grau 1986). Por tanto, dada la ecuación (1.6) resulta:

$$P(t) = e^{Qt} = U\, diag\{\exp(\lambda_1 t), \exp(\lambda_2 t), \exp(\lambda_3 t), \exp(\lambda_4 t)\}\, U^{-1} \tag{1.9}$$

Los λ_i (i = 1,2,3,4) son valores propios de la matriz *Q*. Las columnas de *U* y las filas de U^{-1} son vectores propios derechos e izquierdos respectivamente de *Q*. El lector puede consultar en un libro texto de álgebra lineal, como se calculan los valores propios y los vectores propios de una matriz, por ejemplo, (Schott 1997, Capítulo 3). Lo mismo podría decirse con 5 o más bases.

Cuando *t* incrementa de 0 a ∞, los elementos en la diagonal $p_{jj}(t)$ de la matriz *P(t)* anterior decrecen de 1 a π_j, mientras que los elementos no diagonales $p_{ij}(t)$ incrementan de 0 a π_j, con $p_{ij}(\infty) = \pi_j$, siempre conociendo que la distribución $(\pi_A, \pi_C, \pi_G, \pi_T)$ es la estacionaria.

La mutación puede ocurrir en cada uno de los sitios de la secuencia ADN, teniendo en cuenta que en los mismos, la matriz de probabilidades de transición se calcula una sola vez. En caso de existir un ancestro de la secuencia actualmente procesada, la distribución de probabilidades de un nucleótido en un sitio dado se calcula de manera que la misma se ajusta con la del ancestro en el mismo sitio; de lo contrario, se toma, en la matriz de probabilidades de transición obtenida, la fila correspondiente a esa base y considerándola como su distribución de probabilidades. Un nuevo nucleótido es generado aleatoriamente y se toma como el de sustitución si la distribución de probabilidad correspondiente a él es menor que un valor aleatorio generado. A este mecanismo de generación de bases, que se ha conocido tanto en la simulación como en el campo matemático, se le llama *función de Roulette*.

1.4.3 Modelos markovianos de la evolución molecular

Se han propuesto en la literatura diversos modelos de Markov para la evolución de las secuencias y, se han desarrollado también, varias técnicas para estimar experimentalmente los parámetros de dichos modelos de Markov mediante el análisis de secuencias reales observadas.

La matriz de tasas más general de un modelo de Markov tiene solo el requisito de que los elementos no diagonales son positivos y la suma de cada fila es cero. Se han propuesto varios criterios paramétricos para simplificar la caracterización de la matriz de tasas en los modelos de la evolución de las secuencias ADN. En esta sección se presenta un resumen de los modelos evolutivos de Markov más conocidos en la literatura y relacionados con este trabajo.

1.4.3.1 El modelo Jukes-Cantor (JC69)

El modelo paramétrico más simple de sustitución de nucleótidos fue propuesto por Jukes y Cantor en 1969. Su modelo asume que todos los nucleótidos tienen la misma tasa de sustitución, esto es, dado que la sustitución se ha producido, cualquier otro nucleótido tiene la

misma probabilidad de ser el nucleótido de sustitución. Por tanto, la matriz de tasas de este modelo puede ser parametrizada de la siguiente forma:

$$Q = \begin{bmatrix} -3\alpha & \alpha & \alpha & \alpha \\ \alpha & -3\alpha & \alpha & \alpha \\ \alpha & \alpha & -3\alpha & \alpha \\ \alpha & \alpha & \alpha & -3\alpha \end{bmatrix}, \tag{1.10}$$

donde los nucleótidos son ordenados A, C, G y T. Todos los elementos de la diagonal son iguales a -3α pues que la suma de cada fila de Q debe ser igual a cero. Este modelo asume que la distribución estacionaria de las bases es uniforme siendo $\pi_A = \pi_C = \pi_G = \pi_T = 1/4$, y que el modelo es reversible. Como se ha esbozado anteriormente, una cadena de Markov se dice que es reversible en el tiempo si y sólo si se satisface la siguiente ecuación de balance:

$$\pi_i q_{ij} = \pi_j q_{ji} \qquad \text{para todo } i \neq j, \tag{1.11}$$

donde π_i es la proporción de tiempo que la cadena de Markov se queda en el estado i, $\pi_i q_{ij}$ es el flujo del estado i al j, mientras que $\pi_j q_{ji}$ es el flujo en la dirección opuesta. Esta ecuación significa que el flujo entre cualquier dos estados en la dirección opuesta, es el mismo que la directa (Yang 2006). Usando las tasas de la matriz Q se puede obtener la probabilidad de sustitución de cada nucleótido en un tiempo $t > 0$ mediante la creación de una matriz simple de probabilidades de transición, que en este modelo resulta:

$$P(t) = e^{Qt} = \begin{bmatrix} p_0(t) & p_1(t) & p_1(t) & p_1(t) \\ p_1(t) & p_0(t) & p_1(t) & p_1(t) \\ p_1(t) & p_1(t) & p_0(t) & p_1(t) \\ p_1(t) & p_1(t) & p_1(t) & p_0(t) \end{bmatrix} \quad \text{con} \quad \begin{cases} p_0(t) = \frac{1}{4} + \frac{3}{4} e^{-4\lambda t} \\ p_1(t) = \frac{1}{4} - \frac{1}{4} e^{-4\lambda t} \end{cases} \tag{1.12}$$

1.4.3.2 El modelo Kimura (K80)

Kimura introdujo un modelo un poco más complejo de la evolución de nucleótidos que permite las diferencias entre las tasas de transición y transversión. Como se ha dicho en la sección 1.3.1 anterior, una transición no es más que una sustitución de una purina por otra

purina, o de una pirimidina por otra pirimidina, mientras que una transversión es la sustitución de una purina por una pirimidina o viceversa. Los nucleótidos A y G son purinas, y los nucleótidos C y T son pirimidinas. Transiciones y tranversiones se difieran, ya que las purinas y pirimidinas tienen diferentes estructuras moleculares. La matriz de tasas de sustitución de este modelo es parametrizada como:

$$Q = \begin{bmatrix} -(\alpha+2\beta) & \alpha & \beta & \beta \\ \alpha & -(\alpha+2\beta) & \beta & \beta \\ \beta & \beta & -(\alpha+2\beta) & \alpha \\ \beta & \beta & \alpha & -(\alpha+2\beta) \end{bmatrix} \tag{1.13}$$

El parámetro α controla la tasa de transiciones, mientras que el parámetro β controla la tasa de transversiones. Al igual que el modelo Jukes-Cantor, el modelo Kimura asume también una distribución de frecuencias uniforme de las bases y es un modelo reversible en el tiempo. Este modelo generaliza al modelo Jukes-Cantor (concuerda con él cuando $\alpha=\beta$. La matriz de probabilidades de transición es:

$$P(t) = e^{Qt} = \begin{bmatrix} p_0(t) & p_1(t) & p_2(t) & p_2(t) \\ p_1(t) & p_0(t) & p_2(t) & p_2(t) \\ p_2(t) & p_2(t) & p_0(t) & p_1(t) \\ p_2(t) & p_2(t) & p_1(t) & p_0(t) \end{bmatrix} \tag{1.14}$$

donde los tres elementos diferentes del modelo son:

$$\begin{cases} p_0(t) = \frac{1}{4} + \frac{1}{4}e^{-4\beta t} + \frac{1}{2}e^{-2(\alpha+\beta)t} \\ p_1(t) = \frac{1}{4} + \frac{1}{4}e^{-4\beta t} - \frac{1}{2}e^{-2(\alpha+\beta)t} \\ p_2(t) = \frac{1}{4} - \frac{1}{4}e^{-4\beta t} \end{cases} \tag{1.15}$$

1.4.3.3 El modelo Felsenstein (F81)

Felsenstein introdujo una generalización alternativa del modelo Jukes-Cantor. Este modelo especifica que las tasas de sustitución son proporcionales a las distribuciones estacionarias de los nucleótidos de reemplazo. La matriz de tasas se parametriza de la siguiente forma:

$$Q = \begin{bmatrix} -\alpha(\pi_C + \pi_G + \pi_T) & \alpha\pi_C & \alpha\pi_G & \alpha\pi_T \\ \alpha\pi_A & -\alpha(\pi_A + \pi_G + \pi_T) & \alpha\pi_G & \alpha\pi_T \\ \alpha\pi_A & \alpha\pi_C & -\alpha(\pi_A + \pi_C + \pi_T) & \alpha\pi_T \\ \alpha\pi_A & \alpha\pi_C & \alpha\pi_G & -\alpha(\pi_A + \pi_C + \pi_G) \end{bmatrix} \tag{1.16}$$

Los parámetros π_A, π_C, π_G y π_T especifican la distribución estacionaria en la cual se supone que las frecuencias de las bases son diferentes. Dicho modelo asume la misma tasa de sustitución (α) para todos los nucleótidos y generaliza al modelo Jukes-Cantor en el cual se particulariza que $\pi_A = \pi_C = \pi_C = \pi_T = 1/4$. Al igual que Jukes-Cantor, este modelo es también reversible en el tiempo.

1.4.3.4 El modelo Hasegawa-Kishino-Yano (HKY85)

Hasegawa introdujo un modelo que combina los aspectos de los dos modelos de Jukes-Cantor y Felsenstein. Este modelo es llamado HKY y se puede describir como modelo Felsenstein con un parámetro extra para caracterizar la diferencia entre las transiciones y transversiones. La matriz de tasas de sustitución se puede escribe como:

$$Q = \begin{bmatrix} -\alpha\pi_C - \beta(\pi_G + \pi_T) & \alpha\pi_C & \beta\pi_G & \beta\pi_T \\ \alpha\pi_A & -\alpha\pi_A - \beta(\pi_G + \pi_T) & \beta\pi_G & \beta\pi_T \\ \beta\pi_A & \beta\pi_C & -\alpha\pi_T - \beta(\pi_A + \pi_C) & \alpha\pi_T \\ \beta\pi_A & \beta\pi_C & \alpha\pi_G & -\alpha\pi_G - \beta(\pi_A + \pi_C) \end{bmatrix} \tag{1.17}$$

El modelo HKY generaliza al modelo Kimura cuando $\pi_A = \pi_C = \pi_G = \pi_T = 1/4$ y al modelo Felsenstein cuando $\alpha = \beta$. Como el modelo Felsenstein, el modelo HKY permite una distribución estacionaria no informe y es reversible en el tiempo.

1.4.3.5 El modelo Tamura-Nei (TN93)

Tamura y Nei propusieron un nuevo modelo, que se considera "ciencia constituida" por su actual uso en investigaciones teóricas y en notables productos de software de Bioinformática, el cual incluye los parámetros que distinguen las tasas de sustitución entre purinas y pirimidinas. Este modelo se conoce como una extensión del modelo HKY85 en que ya se introducen dos parámetros α_1 y α_2 en lugar de un simple parámetro presente en el modelo KHY85 para las tasas transicionales. Formalmente, la matriz de tasas de sustitución Q para este modelo, es por tanto, muy similar a la del modelo KHY85, con las diferencias acentuadas. Por tanto como la matriz de probabilidades de transición P se especifica:

$$Q = \begin{bmatrix} -(\alpha_2\pi_G + \beta\pi_Y) & \beta\pi_C & \alpha_2\pi_G & \beta\pi_T \\ \beta\pi_A & -(\alpha_1\pi_T + \beta\pi_R) & \beta\pi_G & \alpha_1\pi_T \\ \alpha_2\pi_A & \beta\pi_C & -(\alpha_2\pi_A + \beta\pi_Y) & \beta\pi_T \\ \beta\pi_A & \alpha_1\pi_C & \beta\pi_G & -(\alpha_1\pi_C + \beta\pi_R) \end{bmatrix} \quad (1.18)$$

donde π_A, π_C, π_G y π_T son, como siempre, las frecuencias estacionarias de los nucleótidos A, C, G y T respectivamente, mientras que $\pi_R = \pi_A + \pi_G$ y $\pi_Y = \pi_C + \pi_T$ son las frecuencias de purinas y pirimidinas. Los parámetros α_1, α_2 y β son, respectivamente, la tasa de transiciones entre pirimidinas, entre purinas y la tasa de transversiones. Es en este modelo, en el que queda formalmente más claro, que a partir de esta matriz Q, es posible calcular la matriz de probabilidades de transición en el tiempo $P_{ij}(t)$ de la cadena, la cual se deriva de la siguiente ecuación diferencial:

$$\frac{dP(t)}{dt} = P(t)Q_r \quad (1.19)$$

A partir de la solución de esta ecuación con condición inicial $P(0)=I$, resulta la matriz de probabilidades de transición:

$$P(t) = e^{tQ_r} \quad (1.20)$$

y sus elementos pueden ser calculados por medio de la descomposición espectral que se ha descrito anteriormente en la sección 1.4.2.

1.4.3.6 El modelo FB o Sánchez - Grau (SG09)

Este modelo es una extensión del modelo TN93 propuesta por Robersy Sánchez y Ricardo Grau, que da lugar al "modelo de cinco bases" (Sánchez and Grau 2009). La adición de una nueva base (D) al conjunto de cuatro ya existentes (A, C, G y T) produce varias modificaciones al modelo TN93 antes expuesto. Esta quinta base también puede ser usada para representar mutaciones *in-del* al ubicarse en los espacios o gaps generados por el alineamiento de secuencias, lo que permite que se utilice la información presente en esos sitios y no se deseche como es inevitable hacer al usar los modelos clásicos.

Como en el modelo TN93, en el presente modelo la tasa de transición entre purinas (α_R) puede ser diferente de aquella entre pirimidinas (α_Y), la tasa de transversiones (β) es la misma para todas las combinaciones de purinas y pirimidinas aunque puede ser distinta de α_1 y α_2 y se permite que las cinco bases posean distintas frecuencias. Dos nuevos parámetros γ y ND están relacionados con la quinta base y no aparecen en el modelo TN93, estos son, respectivamente, la tasa de sustitución entre la D y las otras cuatro y la correspondiente proporción esperada de sitios mostrando estas sustituciones (Sánchez and Grau 2009).

La matriz de las tasas de sustitución para este modelo es 5 x 5:

$$Q = \begin{bmatrix} * & \beta\pi_C & \alpha_R\pi_G & \beta\pi_T & \gamma\pi_D \\ \beta\pi_A & * & \beta\pi_G & \alpha_Y\pi_T & \gamma\pi_D \\ \alpha_R\pi_A & \beta\pi_C & * & \beta\pi_T & \gamma\pi_D \\ \beta\pi_A & \alpha_Y\pi_C & \beta\pi_G & * & \gamma\pi_D \\ \gamma\pi_A & \gamma\pi_C & \gamma\pi_G & \gamma\pi_T & * \end{bmatrix} \tag{1.21}$$

La matriz de probabilidad de transición en el tiempo $P_{ij}(t)$ se deriva, como en el modelo TN93, de la ecuación diferencial:

$$\frac{dP(t)}{dt} = P(t)Q_r \tag{1.22}$$

cuya solución, con condición inicial $P(0)=I$, es:

$$P(t) = e^{Q_r t} \tag{1.23}$$

1.4.3.7 El modelo General Reversible en el Tiempo (GTR)

El modelo GTR sólo asume que el proceso de sustitución de nucleótidos es reversible. Los modelos que se han discutido anteriormente incluyendo JC69 (Jukes-Cantor 1969), K80 (Kimura 1980), F81 (Felsenstein 1981) , HKY85 (Hasegawa-Kishino-Yano 1985) y TN93 (Tamura y Nei 1993) todos son generalizadas con el modelo GTR, y también son reversibles en el tiempo. Incluso SG09 sigue esta filosofía de reversibilidad con la particularidad de que trata con 5 bases en lugar de 4 como todos los mencionados.

Otra condición equivalente para la reversibilidad es que la matriz de tasas de sustitución puede ser escrita como el producto de una matriz simétrica multiplicada por una matriz diagonal. Luego, los elementos en la diagonal especifican las frecuencias de equilibro. De hecho, la matriz de tasas de sustitución para el modelo general reversible en el tiempo es:

$$Q = \begin{bmatrix} * & a\pi_C & b\pi_G & c\pi_T \\ a\pi_A & * & d\pi_G & e\pi_T \\ b\pi_A & d\pi_C & * & f\pi_T \\ c\pi_A & e\pi_C & f\pi_G & * \end{bmatrix} =$$

$$= \begin{bmatrix} * & a & b & c \\ a & * & d & e \\ b & d & * & f \\ c & e & f & * \end{bmatrix} \begin{bmatrix} \pi_A & 0 & 0 & 0 \\ 0 & \pi_C & 0 & 0 \\ 0 & 0 & \pi_G & 0 \\ 0 & 0 & 0 & \pi_T \end{bmatrix} \tag{1.24}$$

Los elementos diagonales de Q son determinados por el requisito de que cada fila de Q tiene suma 0. Esta matriz incluye 10 parámetros: las tasas *a, b, c, d, e, f* y 4 parámetros de frecuencias π_A, π_C, π_G y π_T.

Si se especifican todos los parámetros de la matriz de tasas de sustitución Q, se deriva el modelo GTR. Sin embargo, es posible reducir el número de parámetros cuando los mismos son desconocidos y, se plantea la necesidad de estimarlos a partir de los datos. Esto se puede obtener más fácilmente introduciendo las restricciones que reflejan algunas simetrías aproximadas en el proceso de sustitución (Marco S. 1998). Las restricciones consisten en la

definición de los tipos de cambios de nucleótidos, los cuales se clasifican en dos grupos principales: las transiciones (Ts) y las tranversiones (Tv). Además, es posible distinguir entre sustituciones de purinas y pirimidinas. Una vez que se asumen estas restricciones, sólo quedan dos parámetros independientes: la razón *k* de tasas de Ts y Tv, y la razón γ de dos tipos de tasas de transición. Esto fue lo que definió el modelo TN93. En el caso que $\gamma = 1$, o sea, las transiciones de purinas y pirimidinas tienen la misma tasa, este modelo se reduce al HKY85. Si las frecuencias de las bases son uniformes, es decir $\pi_i = \frac{1}{4}$, entonces el modelo HKY85 se convierte al K80. Para $k = 1$, el HKY85 se reduce al F81, y el K80 deriva al JC69. La jerarquía de los modelos de sustitución discutidos anteriormente se muestra en la *Figura 1.3* siguiente.

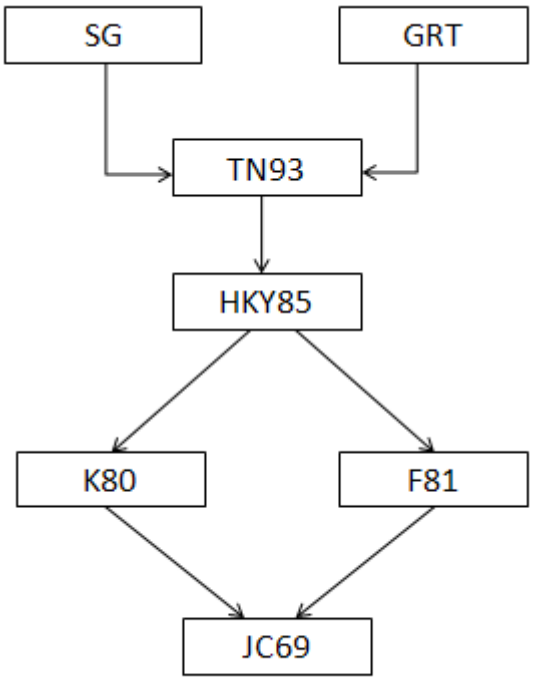


Figure 1.3. *Jerarquía de los modelos de sustitución de nucleótidos*

Por el objetivo de simplificar el uso de los modelos evolutivos, en este trabajo se utilizan solamente los 6 primeros modelos y no se usa el GRT, debido a que este ultimo contiene muchos parámetros y, por tanto, la estimación de los mismos lleva un proceso muy complejo

que, hasta ahora, en la literatura no hemos encontrado ninguna forma adecuada y más sencilla para realizar esta tarea.

1.4.4 Análisis filogenético y estimación de tasas de sustitución por sitio

Los modelos evolutivos de Markov proporcionan una base para hacer referencias filogenéticas, haciéndolas útiles en varias áreas de investigación en Bioinformática. Una de las aplicaciones de estos modelos es la reconstrucción del árbol filogenético, procedimiento que se realiza una vez que se han alineado las secuencias y se ha seleccionado el modelo probabilístico a utilizar. En esta sección se describen los conceptos principales relacionados con la reconstrucción de árboles filogenéticos y se presentan algunos de los algoritmos más usados para construir dichos árboles. A continuación, se describe el procedimiento realizado para la estimación de las tasas de sustitución por sitio a partir de un alineamiento de secuencias, teniendo en cuenta el árbol filogenético del mismo, que se toma como dato de entrada del algoritmo.

1.4.4.1 Construcción de árboles filogenéticos

En los estudios filogenéticos, las relaciones evolutivas entre un grupo de organismos son ilustradas por un árbol filogenético (especie de "dendrograma" como los que aparecen en las técnicas de clusterización). En esencia, un árbol filogenético es un grafo compuesto de nodos y ramas, en el cual una rama solamente conecta a dos nodos adyacentes. Los nodos representan las unidades taxonómicas que pueden ser especies, poblaciones, individuos o genes, mientras que las ramas definen las relaciones entre las unidades taxonómicas en términos de descendente y ancestro (Dan Graur 2000). Un árbol que se obtiene usando un conjunto de datos y un método de reconstrucción de árbol es llamado *árbol inferido.*

Inferir un árbol filogenético, que describa las relaciones evolutivas entre un conjunto de secuencias, ha sido durante mucho tiempo, un tema de máximo interés en Bioinformática y también es útil en la predicción de la funcionalidad del gen. La construcción de un árbol filogenético para un determinado conjunto de secuencias implica en primer lugar, el cálculo de la distancia evolutiva entre cada par de secuencias y, a continuación, especificar la topología del árbol (Keyuan X. 2005). Las distancias evolutivas son generalmente calculadas utilizando técnicas de máxima verosimilitud con modelos de evolución de secuencias. En el capítulo 3,

de Ewens y Grant (W.J. Ewens 2001) se muestra cómo las medidas de máxima verosimilitud de la distancia, se pueden estimar cuando se asumen modelos de Markov paramétricos simples de la evolución. Las distancias evolutivas entre las secuencias se pasan luego como entradas, a los algoritmos que determinan la topología de un árbol filogenético.

Existe una gran variedad de métodos para lograr este objetivo. La inmensa mayoría de ellos se basa en encontrar el árbol que maximice algún criterio de optimalidad (Kumar S. 2008). Los criterios más usados en la reconstrucción de árboles filogenéticos son el de mínima evolución (Saitou 1987), el de máxima parsimonia (Fitch 1971) y el de máxima verosimilitud (Felsenstein 1981).

En este trabajo se usó computacionalmente el FastME (Desper 2002), un producto de software disponible en internet en http://www.atgc-montpellier.fr/fastme/binaries.php, que permite el uso de los algoritmos basados en el criterio mencionado de mínima evolución.

1.4.4.2 Métodos de estimación de las tasas de sustitución por sitio

En la primera parte de esta sección, se presentará el método bayesiano empírico en el cual se hace uso de la distribución gamma, la cual depende de dos parámetros que deben ser estimados con anterioridad. En la segunda parte, se tratará el método de máxima verosimilitud, que permite calcular una primera aproximación de las tasas de sustitución, a partir de la cual se pueden calcular los parámetros α y β de la distribución gamma, necesaria para la estimación final de las tasas.

El método bayesiano empírico

La estimación de las tasas de sustitución por sitio resulta imprescindible cuando se desean conocer las secuencias más probables que tenían los ancestros de los que descienden los individuos actuales, o las que probablemente tendrán los descendientes de éstos. Comúnmente se asume que las tasa de mutación es una variable aleatoria, de la cual se conoce que sigue aproximadamente una distribución gamma, a partir de lo cual ella puede ser estimada usando la distribución condicional (a posteriori) de la misma, dados los datos en el sitio (Yang 2006):

$$f(r \mid x;\theta) = \frac{f(r \mid \theta) f(x \mid r;\theta)}{f(x \mid \theta)}, \tag{1.25}$$

donde r es la tasa de mutación, x los datos observados en las secuencias y θ los parámetros del modelo.

En particular se puede tomar como estimado de la tasa de mutación por sitio, el valor esperado de esta distribución (Yang Z. 1995):

$$r' = E(r \mid x;\theta) = \int_0^\infty rf(r \mid x;\theta)dr = \frac{\int_0^\infty rf(x \mid r;\theta)f(r)dr}{f(x \mid \theta)}, \tag{1.26}$$

donde el denominador se calcula de la siguiente manera:

$$f(x \mid \theta) = \int_0^\infty f(x \mid r;\theta)f(r)dr \tag{1.27}$$

Sustituyendo (1.27) en (1.26) queda la expresión para la estimación a posteriori de la tasa de mutación en el sitio:

$$r' = \frac{\int_0^\infty rf(x \mid r;\theta)f(r)dr}{\int_0^\infty f(x \mid r;\theta)f(r)dr} \tag{1.28}$$

En esta expresión, *f(x|r;θ)* es la verosimilitud del árbol, y *f(r)* es la densidad probabilística *a priori* de las tasas de mutación, o sea, la función de densidad de la distribución gamma.

Las integrales que aparecen en la expresión (1.28) no suelen tener una solución analítica (en términos de funciones elementales) por lo cual se acostumbra discretizarlas de la siguiente forma:

$$\int_0^\infty rf(x \mid r;\theta)f(r)dr = \sum_j r_j f(x \mid r_j;\theta)P(r = r_j)$$

$$\int_0^\infty f(x \mid r;\theta)f(r)dr = \sum_j f(x \mid r_j;\theta)P(r = r_j),$$

donde r_j son las distintas clases discretas de las tasas de sustitución (Nielsen 2005; Yang 2006).

Como esta expresión solo depende de la topología del árbol, de las longitudes de las ramas y de los nucleótidos presentes en las secuencias en el sitio en cuestión, las tasas obtenidas por este método para dos sitios mostrando los mismos datos (o lo que es lo mismo, mostrando el mismo patrón de nucleótidos) serán idénticas. Este hecho permite que en la práctica, se agrupen los sitios en clases de equivalencia según el patrón de nucleótidos que presentan y

solo se calcule la tasa de mutación para un representante de cada clase, asignándoles a los demás ese mismo valor, logrando así que se disminuya el costo computacional del procedimiento.

El método de máxima verosimilitud

En la sección anterior se explicó cómo se pueden estimar las tasas de sustitución por sitio por medio del método bayesiano empírico, bajo el supuesto de que las mismas siguen aproximadamente la distribución gamma. Esta distribución posee dos parámetros (α y β) que deben ser estimados previamente a la aplicación del procedimiento descrito. En esta sección, se habla del método de máxima verosimilitud, que es otra forma de estimar los valores de las tasas de sustitución. En el presente trabajo se aplica este método para estimar los parámetros de la distribución gamma, o sea la distribución de probabilidad *a priori* de las tasas de mutación.

El concepto de la verosimilitud de un árbol filogenético fue introducido por Joseph Felsenstein en 1981 como un criterio de optimización para ser usado en la reconstrucción del mismo (Felsenstein 1981). Se define verosimilitud de un árbol filogenético como la probabilidad de obtener los datos de las secuencias que se tienen a partir del mismo; como se asume que cada sitio evoluciona independientemente de los demás, se puede calcular la verosimilitud del árbol en cada uno de ellos por separado. Si se asume un árbol fijo y se varía la tasa de mutación, se puede usar este criterio como función objetivo y encontrar los valores de las tasas que maximicen la verosimilitud en cada sitio, normalmente haciendo uso de un método numérico. En este caso se escogió el método de Brent (Brent 1973). El método combina la rápida convergencia a la solución buscada de la interpolación parabólica (convergencia cuadrática) con la robustez del método de la sección de oro, más lento (convergencia lineal); además la aplicación de este algoritmo no requiere el cálculo de las derivadas parciales de la función objetivo, lo que garantiza su aplicabilidad en un mayor número de problemas (Press 1992).

Al igual que en el método bayesiano empírico, la verosimilitud solo depende de la topología del árbol, de las longitudes de las ramas y de los nucleótidos presentes en las secuencias en el sitio en cuestión, por lo que también es posible agrupar los sitios de las secuencias según los patrones de nucleótidos que éstos presentan, para ganar en eficiencia computacional

1.5 Modelo de regresión lineal simple, una herramienta estadística para el análisis relacional de los datos

Con frecuencia, nos modelamos en diversas ciencias, problemas en los cuales una variable, *Y,* se puede explicar a través de una función de la variable *X* (ambas posiblemente vectoriales) y lo representamos mediante la ecuación:

$$Y = f(X) \tag{1.29}$$

Si consideramos que la relación *f,* que liga *Y* con *X*, es lineal, entonces (1.29) se puede escribir así:

$$Y_t = \beta_1 + \beta_2 X_t \tag{1.30}$$

Las relaciones del tipo anterior raramente son exactas, sino que más bien son aproximaciones en las que se pueden haberse omitido muchas variables de importancia secundaria: por ello el modelo incluye un término de perturbación aleatoria, u_t, que refleja todos los factores –distintos de *X* -que influyen sobre la variable endógena, aunque que ninguno de ellos sea relevante individualmente y sobre todo no esté correlacionado con *X*. Con ello, la relación quedaría de la siguiente forma:

$$Y_t = \beta_1 + \beta_2 X_t + u_t \tag{1.31}$$

La expresión anterior refleja una relación lineal que nos permite explicar el comportamiento de una variable *Y* dominada variable explicada (o dependiente), a partir de otra variable *X* que llamaremos variable explicativa (o independiente) y recibe el nombre de regresión lineal simple (Morelisi, 2003).

1.5.1 Construcción del modelo de regresión lineal mediante la obtención de los estimadores

Supongamos ahora que disponemos de *T* observaciones de la variable $Y\ (Y_1, Y_2, \ldots, Y_T)$ y de las correspondientes observaciones de $X\ (X_1, X_2, \ldots, X_T)$. Si hacemos extensiva (1.31) a la relación entre observaciones, tendremos el siguiente conjunto de *T* ecuaciones:

$$Y_1 = \beta_1 + \beta_2 X_1 + u_1$$

$$\begin{aligned} Y_2 &= \beta_1 + \beta_2 X_2 + u_2 \\ \dots \quad & \dots \\ Y_T &= \beta_1 + \beta_2 X_T + u_T \end{aligned} \tag{1.32}$$

El sistema de ecuaciones (1.32) se puede escribir abreviadamente de la forma siguiente:

$$Y_t = \beta_1 + \beta_2 X_t + u_t \qquad t = 1,2,\dots, T \tag{1.33}$$

El objetivo principal de la regresión es la determinación o estimación de β_1 y β_2 a partir de la información contenida en las observaciones que disponemos de esas dos variables *X* e *Y* sobre una muestra de *T* individuos. El primer paso en un análisis de regresión es representar estos datos sobre unos ejes coordenados *x*-*y*. Esta representación es el llamado *diagrama de dispersión* (Gibergans, 2003). Puede ayudar mucho en la búsqueda de un modelo que describa la relación entre las dos variables. Si la relación de dependencia entre *Y* y *X* fuera lineal exacta, las observaciones se situarían a lo largo de una recta (*Figura 1.4*). Pero si la dependencia entre *Y* y *X* es estocástica, entonces, en general, las observaciones no se alinearán a lo largo de una recta, sino que formará una nube de puntos (*Figura 1.5*) pero cercana a una recta. Si designamos mediante β_1' y β_2' las estimaciones de β_1 y β_2, respectivamente, la ordenada de la recta para el valor X_t vendrá dada por:

$$Y_t' = \beta_1' + \beta_2' X_t \tag{1.34}$$

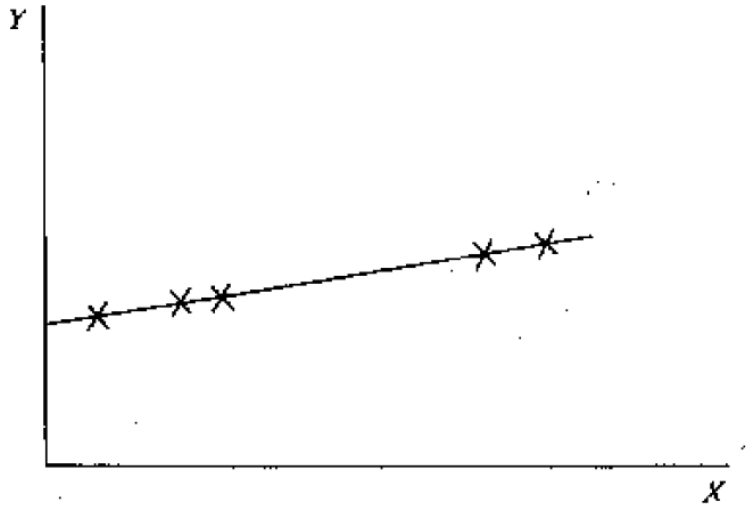


Figura 1.4: *Modelo de regresión lineal exacta*

El problema ahora es hallar unos estimadores β_1' y β_2' tales que la recta que pasa por los puntos *(X_t, Y_t')* se ajuste lo mejor posible a los puntos *(X_t, Y_t)*. Se denomina, error o residuo, a la diferencia entre el valor observado de la variable dependiente y el valor ajustado, es decir:

$$u_t' = Y_t - Y_t' = Y_t - \beta_1' - \beta_2' X_t \tag{1.35}$$

En el modelo de regresión lineal simple hay tres parámetros que se deben estimar: los coeficientes de la recta de regresión, β_1 y β_2 ; y la varianza de la distribución normal, σ^2. El cálculo de estimadores para estos parámetros puede hacerse por diferentes métodos, siendo el más utilizado el método de mínimos cuadrados. Este método consiste en buscar los valores de los parámetros β_1' y β_2' de manera que la suma de los cuadrados de los residuos (S) sea mínima.

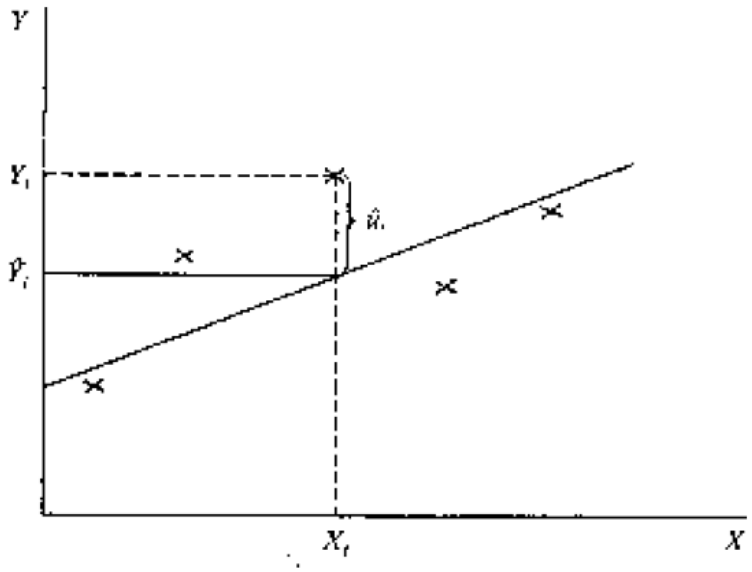


Figura 1.5: *Modelo de regresión lineal simple no exacta*

Esta recta es la recta de regresión por mínimos cuadrados. Siendo la suma de los cuadrados la expresión:

$$S = \sum_{t=1}^{T} (Y_t - \beta_1' - \beta_2' X_t)^2 \tag{1.36}$$

Para minimizar *S*, derivamos parcialmente respecto a β_1' y β_2':

$$\frac{\partial S}{\partial \beta_1'} = -2\sum_{t=1}^{T} (Y_t - \beta_1' - \beta_2' X_t) \tag{1.37}$$

$$\frac{\partial S}{\partial \beta_2'} = -2\sum_{t=1}^{T} (Y_t - \beta_1' - \beta_2' X_t) X_t \tag{1.38}$$

y las igualamos a cero. Así obtenemos el sistema de ecuaciones siguiente, conocido como s*istema de ecuaciones normales* de la recta de regresión:

$$\begin{cases} \sum_{t=1}^{T} (Y_t - \beta_1' - \beta_2' X_t) = 0 \\ \sum_{t=1}^{T} (Y_t - \beta_1' - \beta_2' X_t) X_t = 0 \end{cases} \tag{1.39}$$

Resolviendo este sistema, se obtiene de forma inmediata, el estimador de β_2':

$$\beta_2' = \frac{\sum_{t=1}^{T}(Y_t - \overline{Y})(X_t - \overline{X})}{\sum_{t=1}^{T}(X_t - \overline{X})^2} \tag{1.40}$$

y $$\beta_1' = \overline{Y}_2 \overline{X} \tag{1.41}$$

Dividiendo el numerador y el denominador de (1.40) por T se obtiene:

$$\beta_2' = \frac{\frac{\sum_{t=1}^{T}(Y_t - \overline{Y})(X_t - \overline{X})}{T}}{\frac{\sum_{t=1}^{T}(X_t - \overline{X})^2}{T}} = \frac{\text{cov}(X,Y)}{\text{var}(X)} \tag{1.42}$$

De acuerdo con (1.42), la estimación de β_2' se obtiene dividiendo la covarianza muestral de X e Y por la varianza muestral de X. Dado que la varianza de X no puede ser negativa, el signo de β_2' será el mismo que el de la covarianza muestral de X e Y.

La recta de regresión ahora se puede escribir de la manera siguiente:

$$Y' = \beta_1' + \beta_2' X \tag{1.43}$$

Los residuos calculados con la recta de regresión los llamaremos e_i, es decir:

$$e_i = Y_i - Y_i', \tag{1.44}$$

Donde Y_i' es el valor estimado para la recta de regresión.

1.5.2 Los residuos en la estadística

En la regresión lineal, los residuos o residuales son utilizados para verificar la adecuación del modelo. Los residuos expresan la diferencia entre una observación y su valor ajustado. Estos pueden ser usados para evaluar la adecuación del ajuste de un modelo, con respecto a la elección de la función de varianza, la función enlace y en términos del predictor lineal (Gibergans, 2003).

Los residuos $e_i = Y_i - Y_i'$ de un modelo de regresión lineal deben ser variables aleatorias con distribución $e_i \sim N(0, \sigma^2(1-h_{ii}))$, i=1,…, n siendo n el tamaño de la muestra de los datos. Existen tres variantes de residuos, los cuales son:

1. Residuos crudos: Es la diferencia entre el valor observado y el valor predictivo, tal y como se ha dado en la fórmula (1.44) anterior.

2. Residuos estandarizados: Son aquellos que se obtienen dividiendo los residuos crudos por la estimación del error estándar y se definen como:

$$r_i = \frac{e_i}{s_R\sqrt{1-v_{ii}}} \qquad \text{(i=1,…, n)} \tag{1.45}$$

3. Residuos estudentizados: son los que se resultan de la división de los residuos-fila por la estimación de su desviación estándar y se definen como:

$$t_i = \frac{e_i}{s_{R(i)}\sqrt{1-v_{ii}}} \qquad \text{(i=1,…, n)} \tag{1.46}$$

Para chequear el modelo de regresión se deben utilizar los residuos estandarizados o los estudentizados. Típicamente, las desviaciones estándares de los residuos en una muestra varían considerablemente de un punto de datos al otro aunque los errores todos tienen igual la desviación estándar, particularmente en el análisis de regresión. Por tanto, no es conveniente comparar los residuos en diferentes puntos de datos sin primero estudentizarlos. Esta técnica es muy importante para la detección de los outliers, o datos atípicos, los cuales se definen como observaciones con residuos estandarizados grandes ($|r_i| > 2$).

Nos interesa ahora de cómo se estudentizan los residuos de un modelo de regresión lineal simple. Para ello, partimos de una matriz del tamaño (n x 2) siguiente:

$$X = \begin{bmatrix} 1 & x_1 \\ \dots & \dots \\ 1 & x_n \end{bmatrix} \tag{1.47}$$

Donde x_i (i=1,…, n) son los datos de la muestra. La matriz "hat" H es la proyección ortogonal sobre el espacio de columna de la matriz X y se obtiene por:

$$H = X(X^T X)^{-1} X^T \tag{1.48}$$

La varianza del *i*-ésimo residuo es:

$$\text{var}(\hat{\varepsilon}_i) = \sigma^2(1 - h_{ii}) \tag{1.49}$$

Siendo h_{ii} el i-ésimo de la matriz H. Luego, el residuo estudentizado correspondiente resulta por:

$$\frac{\hat{\varepsilon}_i}{\hat{\sigma}\sqrt{1-h_{ii}}} \tag{1.50}$$

Donde $\hat{\sigma}$ es la estimación apropiada de σ que se calcula de la siguiente forma:

$$\hat{\sigma}^2 = \frac{1}{n-m}\sum_{j=1}^{n}\hat{\varepsilon}_j^{\,2}, \tag{1.51}$$

Sabiendo que *m* es el número de los parámetros en el modelo (*m* es igual a 2 para el modelo simple). Sin embargo, es deseado excluir la i-ésima observación del proceso de estimación de las varianzas cuando se considera que el i-ésimo caso pudiera ser un outlier. Consecuentemente, $\hat{\sigma}$ se puede estimar con la forma:

$$\hat{\sigma}_{(i)}^{\,2} = \frac{1}{n-m-1}\sum_{\substack{j=1 \\ j\neq i}}^{n}\hat{\varepsilon}_j^{\,2}, \tag{1.52}$$

que basa en todos los casos excepto el i-ésimo. Si se usa la estimación (1.52) , excluyendo un caso atípico, entonces el residuo se dice que es estudentizado externamente; de lo contrario, es decir, si usamos la forma (1.51) para estimar la varianza σ estamos estudentizando el residuo internamente.

1.6 Diseño y análisis de experimentos computacionales de la simulación

Los modelos de simulación computacional, o Monte Carlo, no son resueltos por el análisis matemático, pero son usados para la experimentación numérica. Estos experimentos son significativos para responder a las preguntas de interés sobre el mundo real, es decir que el hombre pudiera usar su modelo de simulación para responder las preguntas de tipo "Que… Si…" como: ¿Qué pasará si las entradas del modelo de la simulación cambiaran? Esto es llamado análisis de sensibilidad y es guiado por la teoría estadística en el diseño de experimentos (DOE por sus siglas en inglés) (Gentle et al., 2004).

Diseño experimental es una técnica estadística para organizar y analizar los experimentos. Prácticamente corremos un programa de simulación dado con diferentes combinaciones de las

entradas, estas combinaciones son llamados escenarios y las entradas son llamadas factores del DOE, que pudieran ser un parámetro, una variable de entrada, o un módulo del modelo de la simulación (Kleijnen 2001). En general, el DOE puede ser definido como la selección de la combinación de los factores que pueden ser experimentados con un modelo de simulación. Dichos factores son las variables independientes que se controlan por el investigador de forma tal que se pueden modificar sus valores, y las salidas son las variables respuestas dependientes que representan el fenómeno de interés.

Las herramientas estadísticas utilizadas para analizar los datos que resultan de un experimento son ampliamente conocidas y muchas de ellas se basan en el análisis de varianza (ANOVA) y en la regresión múltiple o modelo lineal general aunque también, y de manera más amplia, en los modelos lineales generalizados; en caso de ser necesario, es posible utilizar métodos no paramétricos equivalentes (Salazar 2008).

Obsérvese que un escenario en particular se forma de la combinación de los diferentes niveles de los factores. Cuando dicho escenario se corresponde con una situación virtual, propiciada por algún software, se habla de estudios de simulación. Dichos estudios son muy comunes en la investigación estadística (Robert and Casella 2004); un ejemplo de ello es cuando se quiere estudiar el efecto del *tamaño muestral* en la estimación de cierto parámetro. Normalmente el análisis de los datos que resultan de un estudio de simulación se limita a la descripción de los mismos por medio de gráficos y tablas que contienen medidas de tendencia central y dispersión de la variable(s) respuesta(s) para cada uno de los diferentes escenarios simulados, sin pasar luego por un análisis inferencial.

Una de las diferencias fundamentales entre este tipo de experimentos computacionales y los experimentos físicos es que en los primeros se tiene mayor control sobre los factores que inciden sobre una respuesta ya que estos factores son generados a partir de algoritmos estocásticos predefinidos cuyos parámetros son especificados por el investigador. En contraste, los experimentos físicos generan resultados que dependen de factores que frecuentemente no es posible controlar (Salazar 2008).

Como lo explica Montgomery en su libro de diseño de experimentos (Montgomery and Douglas 1997), para utilizar un enfoque estadístico en el diseño y análisis de un experimento es necesario que todas las personas involucradas en el proceso entiendan de qué se trata el

problema, qué es lo que exactamente se va a estudiar, cómo se recolectarán los datos y tener una idea del análisis cuantitativo que se llevará a cabo. Lo mismo ocurriría en un estudio de simulación; lo más importante que hay que definir es qué se va a medir (variable(s) respuesta(s)) y en función de qué (factores y bloques). Montgomery presenta una guía sencilla para diseñar un experimento que consiste en siete pasos: 1) planteamiento del problema, 2) selección de la(s) variable(s) respuesta, 3) elección de factores y niveles, 4) elección del diseño experimental o tipo de experimento, 5) desarrollo del experimento, 6) análisis estadístico de los datos, y 7) conclusiones.

El planteamiento del problema lleva a deducir cuáles variables respuesta serán medidas, y a su vez, según la escala de estas variables, qué tipo de análisis elegir. Los factores y sus niveles, tienen que ver también con el planteamiento del problema ya que de éstos van a depender las variables respuesta. La idea general es observar el comportamiento de la(s) variable(s) respuesta en función de los factores y estimar el efecto de estos últimos sobre la respuesta. En los estudios de simulación es fácil averiguar cuáles podrían ser las variables respuesta, los factores y las variables controlables. Variables respuesta en muchos estudios de simulación podrían ser por ejemplo el sesgo relativo, una media, una medida de dispersión o una proporción de aciertos. Como factores se tomarían aquellas variables para las cuales se desea ver el efecto sobre la respuesta. Ejemplos de factores podrían ser el tipo de distribución, el tamaño de muestra, etc.

En cuanto al tipo de experimento, en términos generales se piensa en un experimento multifactorial de efectos fijos. Es muy probable que el recurso más importante en los estudios de simulación sea el tiempo invertido en las corridas de las simulaciones y que esto lleve a pensar en diseños experimentales de efectos fijos más especiales como son los diseños 2^k, 3^k y los diseños fraccionados; por supuesto que esto dependería del criterio del investigador. En cualquier caso es útil presentar la caja o esquema del diseño del estudio como se muestra en la *Figura 1.3*. Una vez estén estos aspectos claros, se procede a correr las simulaciones. Al término de éstas se construye una base de datos con los resultados y se procede al análisis estadístico. Éste comenzaría con los gráficos de cajas y bigotes y en general con el uso de todas las herramientas descriptivas que ayuden a formular posibles hipótesis que serán probadas posteriormente por medio del análisis de varianza, el análisis de regresión o las

comparaciones múltiples, siempre y cuando se cumplan los tres supuestos o requisitos: 1) normalidad, 2) homocedasticidad e 3) independencia de los residuales. En caso de que alguno(s) de estos no se cumplan, habrá que pensar en una prueba o análisis no paramétrico equivalente (Salazar 2008).

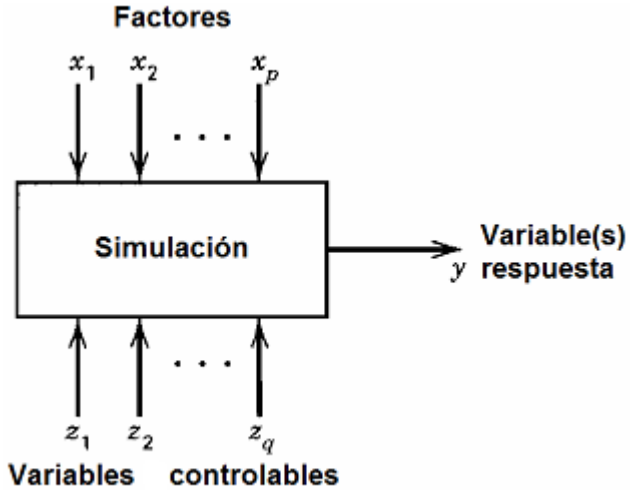


***Figura 1.3**. Esquema general de un estudio de simulación. Adaptación del esquema de un diseño experimental presentado en Montgomery (Montgomery and Douglas 1997).*

1.7 Herramientas computacionales utilizadas

En esta sección se introducen las herramientas utilizadas en la implementación de los algoritmos. Se da una breve reseña histórica del lenguaje de programación C++ y se explican las razones por las que fue seleccionado. También se describen las dos bibliotecas que se utilizaron para los métodos numéricos.

1.7.1 El lenguaje de programación C++

C++ es un lenguaje orientado a objetos derivado del C que fue diseñado a mediados de los años 80 por Bjarne Stroustrup. La intención de su creación fue extender el exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido. Posteriormente se añadieron facilidades de programación genérica, la cual se sumó a los otros dos paradigmas que ya estaban admitidos (programación estructurada y la programación

orientada a objetos, POO). Por esto se suele decir que el C++ es un lenguaje de programación multiparadigma y no es un lenguaje orientado a objetos puro. Más bien se trataba de añadirle "objetos" al clásico **C**, ya que el nuevo paradigma de programación "con objetos", se mostraba como un paso adelante en el arte de la programación. En el diseño de la Librería Estándar C++, se ha usado ampliamente la dualidad de mezcla de un lenguaje tradicional con elementos de POO, lo que ha permitido un modelo muy avanzado de programación extraordinariamente flexible (programación genérica). Desde luego, **C++** es un lenguaje de programación extremadamente largo y complejo, sin embargo, ha experimentado un extraordinario éxito desde su creación. De hecho, muchos sistemas operativos, compiladores e intérpretes han sido escritos en **C++**. Una de las razones de su éxito es ser un lenguaje de propósito general que se adapta a múltiples situaciones. Para comprobar el éxito e importancia de los desarrollos realizados en **C++** se puede visitar la página que mantiene el Dr. Stroustrup al respecto: www.research.att.com.

Como se ha indicado, C++ fue diseñado para mejorar a un lenguaje puramente estructural C añadiendo el nuevo paradigma de programación que usa objetos y sus interacciones, para diseñar aplicaciones y programas de la computadora. El término de Programación Orientada a Objetos indica más una forma de diseño y una metodología de desarrollo de software, que un lenguaje de programación. En las secciones siguientes veremos cómo se organizan los objetos usados y cómo se relacionan ellos en el desarrollo del modelo de la simulación.

1.7.2 La biblioteca BTL

En este trabajo fue necesario aplicar varios métodos numéricos de cierta complejidad, como el cálculo de los valores y vectores propios de una matriz. La implementación de este tipo de algoritmos puede resultar engorrosa y propensa a errores, pero, afortunadamente, ya estos han sido implementados en todas sus variantes y, lo que es más importante, estas implementaciones ya han sido depuradas por sus autores y por los usuarios de las mismas.

En este trabajo se utilizó la biblioteca Bioinformatics Template Library (BTL) para los cálculos numéricos, la cual es un software distribuido libre y gratuitamente bajo el General Public License (GPL) que el lector puede encontrar en el sitio web http://www.bioinformatics.org/project/?group_id=184. Esta biblioteca está escrita en el lenguaje C++ y sigue el estándar y el estilo de la biblioteca Standard Template Library (STL) que forma

parte del ISO/ANSI C++ estándar. El objetivo de BTL se encuentra en las estructuras de datos y algoritmos utilizados en los campos de la bioinformática y la modelación molecular. En esta biblioteca se presenta una nueva aplicación de técnicas de programación genérica en forma de una biblioteca de las componentes de C++ mediante el uso de plantillas para implementar los algoritmos y estructuras de datos que permiten el desarrollo de módulos de programación eficiente proveyendo códigos en forma genética y facilitando el combinarlos flexiblemente con las bibliotecas de clases orientadas a objetos.

Aunque el BTL se ha diseñado con aplicaciones de biología molecular, contiene las clases de más utilidad general, por ejemplo, el contenedor de la matriz y algoritmos asociados se podría utilizar para el álgebra matricial en muchos contextos.

1.7.3 La biblioteca BOOM

Esta biblioteca, denominada así por Bioinformatics Object-Oriented Modules, es una colección de clases reusables para el desarrollo de aplicación y modelación orientadas a objetos eficientes en la Bioinformática. Como la BTL, esta biblioteca también se escribió en el lenguaje C++, es un software distribuido libre y gratuitamente bajo el General Public License (GPL) que el lector podría encontrar en la página http://www.bioinformatics.org/project/?group_id=466.

Se puede encontrar en esta biblioteca un gran número de clases y funciones útiles para la modelación de la biología molecular, la implementación de los métodos numéricos y estadísticos que, entre otras, algunas de las cuales se utilizaron en este trabajo con el objetivo de realizar los cálculos elementales de la estadística de forma más eficiente y llevar a cabo las operaciones necesarias para el análisis de las secuencias genómicas.

En el capítulo 3 especificaremos precisamente cuales funciones de esta biblioteca utilizaremos para la implementación de los algoritmos de la simulación.

1.8 Consideraciones finales del capítulo

Hemos expuesto los conceptos básicos de la evolución molecular, los fundamentos matemáticos de la simulación computacional, en particular el método de Monte Carlo y la aplicación de esta técnica para la simulación de la evolución de las poblaciones virales, en la cual el método de Markov es usado como el núcleo para implementar nuestro algoritmo. Se ha

presentado además, la técnica del análisis de regresión lineal como una herramienta estadística muy útil para estudiar la relación entre los datos, la cual se usa en la implementación del algoritmo que se presentará en el siguiente capítulo.

Finalmente se han abordado las herramientas computacionales que se usan en el trabajo para implementar los algoritmos de la simulación, las cuales incluyen el uso del lenguaje de programación C++, y la aplicación de las bibliotecas estándares para realizar las tareas elementales y complejas que forman parte de los algoritmos.

2 IMPLEMENTACIÓN DE LOS ALGORITMOS PARA LA SIMULACIÓN DE POBLACIONES VIRALES

La técnica de simulación de Markov se ha aplicado extensiva y efectivamente en el área de Bioinformática, especialmente en el campo de la genética evolutiva, en la cual se pueden modelar los procesos del tipo estocástico, como la divergencia genética a lo largo del tiempo. Estos procesos son conocidos como modelos estocásticos de Markov de primer orden, pues tienen la propiedad de que el estado presente sólo depende del inmediato anterior, por ejemplo, la probabilidad de mutación de un nucleótido A que cambia a otro T no se ve afectada por la historia previa que condujo hasta A. En estos modelos, las tasas de mutación entre diferentes estados discretos de la secuencia de ADN se describen mediante la matriz de transición markoviana $Q^*=\{q^*_{ij}\}$. La simulación utiliza esta matriz y procesa en un tiempo T, empezando a partir de un conjunto dado de secuencias semillas. Las secuencias generadas luego pueden ser pasadas como entradas de los métodos que estiman los parámetros del modelo de Markov para evaluar su exactitud en la reconstrucción de Q^*.

En este capítulo, se presentan los cuatro procedimientos usados para la simulación que se basan en la técnica de Markov. Se introducen, además, los métodos de evaluación de la exactitud de los modelos evolutivos en la reconstrucción de la matriz de tasas de sustitución, a lo cual se le sigue una descripción por pasos del proceso evolutivo de las poblaciones virales y el análisis de la complejidad temporal de los algoritmos de la simulación. Por último, se exponen el esquema y el diseño general de la aplicación, los cuales constituyen un paso indispensable para construir exitosamente un sistema integrado capaz de simular eficientemente el proceso evolutivo de las poblaciones virales.

2.1 Procedimientos de simulación basados en la técnica de Markov

Los métodos clásicos de simulación de la evolución de secuencias requieren como entradas un árbol filogenético con la longitud de una rama específica, una secuencia de raíz y un modelo de Markov de la evolución de secuencias. La simulación es realizada por cada rama, empezando por la raíz y terminando en las puntas de las hojas. Para cada rama, se construye una matriz de mutación $P(t)$, a partir de del modelo de Markov propuesto, y es escalada de

manera tal, que el número medio de sustituciones coincide con la longitud de la rama. La matriz *P(t)* luego se aplica en la parte superior de la rama para crear una secuencia en la parte inferior (Nielsen 2005). La ventaja de nuestro simulador es que no requiere como entrada un árbol filogenético, sino que lo genera como parte de la simulación de acuerdo a un proceso que se junta con el proceso de simulación de mutación de secuencias, reflejando así sus dependencias esperadas en la vida real.

Todos los sitios de la secuencia evolucionan de forma independiente, pero son regidos por la misma matriz Q^*. Suponiendo que un sitio dado actualmente está en el estado *i*, es decir, el nucleótido *i* actualmente está en esa posición. Un proceso de Markov de tiempo continuo determina que el tiempo de espera evolutivo entre las sustituciones, o sea, el periodo de tiempo que el sitio mantiene en el estado *i*, es exponencialmente distribuido con el parámetro $|q^*_{ii}|$ (Nielsen 2005). Dado que este último es la tasa de sustitución de un nucleótido determinado *i,* es fácil construir una variable aleatoria de distribución exponencial a partir de la otra uniformemente distribuida, utilizando la siguiente ecuación:

$$t = -\frac{1}{|q^*_{ii}|}\log_e(u), \qquad (2.1)$$

donde *u* es un número aleatorio uniforme entre 0 y 1.

Supongamos que *T* representa el tiempo transcurrido durante cada paso de la simulación, para cada sitio de la secuencia, se genera un número aleatorio uniformemente entre 0 y 1, este número luego es usado para calcular el tiempo de espera *t.* Si $t < T$ entonces ocurre una sustitución en ese sitio, de lo contrario el mismo se mantendrá en su estado actual.

Suponiendo que va a ocurrir una mutación, el nucleótido de reemplazo se selecciona de acuerdo a la cadena de Markov de tiempo discreto $D^* = \{d^*_{ij}\}$ definida por:

$$d^*_{ij} = \begin{cases} 0, & i = j \\ \dfrac{q^*_{ij}}{q^*_{ii}}, & i \neq j \end{cases} \qquad (2.2)$$

El nucleótido de sustitución se simula generando aleatoriamente un número que es uniformemente distribuido entre 0 y 1. Si este número aleatorio es menor que d^*_{i1}, el nucleótido 1 sustituye al nucleótido i. De lo contrario, si el número aleatorio es menor que $d^*_{i1} + d^*_{i2}$, el nucleótido 2 sustituye al i. Este proceso repite hasta que se encuentre el

nucleótido de sustitución. Por el hecho de que $\sum_j d_{ij}^* = 1$, siempre se encuentra un nucleótido de sustitución , y porque $d_{ii}^* = 0$, el nucleótido sustituido no puede ser el mismo (Keyuan X. 2005).

Este proceso de simulación se basa en la siguiente caracterización de una cadena de Markov de tiempo continuo. El tiempo de espera hasta la siguiente transición es exponencial con la media $1/|q_{ii}^*|$. Si ignoramos los tiempos de espera entre las transiciones, la secuencia de los estados visitados por el proceso constituye una cadena de Markov en tiempo discreto, la cual es llamada *la cadena de salto*. La matriz de transición de esa cadena está dada por:

$$M = \begin{bmatrix} 0 & \frac{q_{AC}}{q_A} & \frac{q_{AG}}{q_A} & \frac{q_{AT}}{q_A} \\ \frac{q_{CA}}{q_C} & 0 & \frac{q_{CG}}{q_C} & \frac{q_{CT}}{q_C} \\ \frac{q_{GA}}{q_G} & \frac{q_{GC}}{q_G} & 0 & \frac{q_{GT}}{q_G} \\ \frac{q_{TA}}{q_T} & \frac{q_{TC}}{q_T} & \frac{q_{TG}}{q_T} & 0 \end{bmatrix} \qquad (2.3)$$

El algoritmo de simulación de los tiempos de espera exponenciales y la cadena de salto se puede aplicar a toda la secuencia en lugar de un sitio. La tasa de sustitución total es la suma de las tasas en todos los sitios, y el tiempo de espera hasta que una sustitución ocurra en cualquier sitio de toda la secuencia tiene una distribución exponencial con la media igual a la inversa de la tasa total. Una ventaja de este mecanismo de simulación es que no requiere el cálculo de la matriz de probabilidades *P(t)* sobre el tiempo *t,* ya que tanto los tiempos de espera como la matriz de transición para la cadena de salto son totalmente especificados por la matriz de tasas instantáneas Q^*.

Otro método de simulación llamado Metropolis-Hasting, como se había dicho anteriormente, es conocido como el método más popular de la familia de los métodos Markov Chain Monte Carlo, y comúnmente muy usado para la evolución de secuencias (Andrieu, 2003). Este método consiste en generar un nuevo valor candidato x^* dado el valor actual *x,* de acuerdo a la probabilidad de distribución propuesta $q(x^* \mid x)$. La cadena de Markov se mueve hacia x^* con la probabilidad de aceptación $p(x, x^*) = \min\left\{1, \frac{p(x^*)q(x \mid x^*)}{p(x)q(x^* \mid x)}\right\}$, en la cual *p(x)* es la distribución

invariante definida en *x,* o se queda en este último. El pseudocódigo de este algoritmo se muestra a continuación:

1. Inicializar $x^{(0)}$
2. Para i=0 hasta N-1:
   - Muestrear $u \sim U(0,1)$
   - Muestrear $x^* \sim q(x^* \mid x^{(i)})$
   - Si $u < p(x^{(i)}, x^*) = \min\left\{1, \frac{p(x^*)q(x^{(i)} \mid x^*)}{p(x^{(i)})q(x^* \mid x^{(i)})}\right\}$

     $x^{(i+1)} = x^*$

     si no

     $x^{(i+1)} = x^{(i)}$

Este algoritmo es el más general de los métodos de MCMC. La mayoría de los otros algoritmos prácticos son interpretados como sus casos especiales o extensiones, que difieren en la especificación de las opciones de la probabilidad de distribución propuesta $q(x^* \mid x)$. En este trabajo, se usa el método llamado "*Muestra independiente*", que es una de las instancias más simples de Metropolis-Hasting. Este método consiste en el hecho de que la distribución propuesta es independiente del estado actual, es decir, $q(x^* \mid x^{(i)}) = q(x^*)$ y $q(x^{(i)} \mid x^*) = q(x^{(i)})$. La probabilidad de aceptación está dada por:

$$p(x^{(i)}, x^*) = \min\left\{1, \frac{p(x^*)q(x^{(i)} \mid x^*)}{p(x^{(i)})q(x^* \mid x^{(i)})}\right\}$$

$$= \min\left\{1, \frac{p(x^*)q(x^{(i)})}{p(x^{(i)})q(x^*)}\right\} \qquad (2.4)$$

Una vez que el valor candidato x^* sea aceptado, es decir, ha ocurrido una mutación del nucleótido en el sitio dado, la probabilidad de distribución y la probabilidad de mutación de ese nucleótido se van a determinar para el nuevo estado, de acuerdo a la probabilidad previamente definida para el nuevo nucleótido. En el caso de que no haya cambio ninguno, o sea, la cadena de Markov se mantenga en el estado actual con el mismo valor de nucleótido en el sitio dado, entonces no hay necesidad de cambiar las probabilidades de distribución y

mutación de la base en ese sitio. Este procedimiento de generación de muestras basado en la técnica de Markov para simular la mutación de las secuencias, se analizará con más detalle cuando se expliquen los algoritmos de simulación en las secciones 2.3 y 2.4 de este capítulo. El método de Metropolis-Hasting, la estimación de tiempo evolutivo y la matriz de transición de la cadena de salto, son los posibles mecanismos para la simulación, de modo que los mismos pudieran ser usados de forma combinada entre sí para generar diferentes procedimientos, los cuales pueden aplicarse independientemente para examinar la efectividad de cada uno de ellos en el proceso evolutivo de las secuencias genómicas.

2.1.1 Simulación basada en el método de Metropolis-Hasting y el tiempo de espera

En este procedimiento, la estimación del tiempo de espera evolutivo es el paso realizado antes para evaluar la condición necesaria en la cual se pudiera ocurrir o no una mutación de nucleótido en un sitio determinado. Como se ha expuesto previamente, para simular el tiempo de espera entre dos sustituciones, se genera primero un número aleatorio uniforme entre 0 y 1. Suponiendo que el estado actual del sitio dado es *i,* o sea, el mismo está ocupado por el nucleótido *i,* la tasa de sustitución de este último se puede conocer a partir de la matriz de tasas instantáneas Q^*. El tiempo de espera *t* se calcula utilizando la fórmula (2.1).

Dado el tiempo evolutivo *T* transcurrido en cada paso de la simulación. Si $t<T$, se dice que hay un cambio de nucleótido en el sitio dado. Cuando esto ocurre, aplicamos el método de Metropolis-Hasting para generar el nucleótido de sustitución, de manera tal que el nuevo debe ser diferente que su ancestro. Para ello, se aplica el Metropolis-Hasting de forma repetitiva hasta que un nuevo nucleótido se encuentre.

2.1.2 Simulación basada en el tiempo de espera y la matriz de transición

Como el caso anterior, el proceso evolutivo de secuencias que se simula utilizando este mecanismo considera que el tiempo de espera es una condición necesaria y suficiente para saber si hay mutación de nucleótido en un sitio dado de la secuencia. El proceso de simulación del tiempo de espera se realiza de la misma forma que en el procedimiento anterior. En caso que haya cambio de nucleótido, es decir, el tiempo de espera *t* estimada es menor que el tiempo evolutivo propuesto en la simulación, se buscara luego el nucleótido de reemplazo mediante la matriz de transición. Esta matriz se obtiene a partir de la matriz de tasas de

sustitución instantáneas y tiene la forma mostrada en (2.3). La selección del nuevo nucleótido de reemplazo se hace de la siguiente manera. Suponiendo que el nucleótido de partida ocupado en el sitio dado es *i*, o sea, el estado actual del proceso es *i*, a partir de la matriz de tasas de sustitución Q^*, se obtiene la fila correspondiente a ese nucleótido. Los valores en esta fila constituyen una cadena de Markov de tiempo discreto, la cual se define de forma tal y como se muestra en (2.2). La determinación del nucleótido al cual el proceso cambiara se ha explicado con anterioridad.

2.1.3 Simulación basada en el método de Metropolis-Hasting con repetición

Este procedimiento consiste en utilizar el método Metropolis-Hasting para determinar el cambio de nucleótido en un sitio de la secuencia. Para ello, lo primero que se hace es calcular la matriz de probabilidades de sustitución *P(t),* a partir de la matriz de tasas Q^*. En la matriz *P(t)*, se obtiene la fila correspondiente al nucleótido que actualmente está ocupado en el sitio dado, esta fila no es más que el vector de distribución de probabilidades de ese nucleótido y se denota por *p*. Para saber si este último se muta o no, se procesa de la siguiente forma: se genera un número aleatorio uniforme en (0,1) llamado *u*, y suponiendo que el nucleótido actual es *i.* Si $u < p(i)$, decimos que no hay cambio en el sitio dado. En el caso contrario, se buscara un nucleótido de sustitución tal que este es diferente al nucleótido actual, mediante la aplicación repetitiva del método Metropolis-Hasting. De esta forma, cuando hay mutación se garantiza siempre que el nucleótido encontrado no coincida con el ancestral y el proceso cambia a un nuevo estado.

2.1.4 Simulación basada en el método de Metropolis-Hasting sin repetición

La diferencia de este procedimiento con los otros presentados anteriormente es que el mismo no tiene la condición de examinar si una mutación ocurre o no en un sitio determinado. El método de Metropolis-Hasting se aplica una sola vez para cada sitio de la secuencia, teniendo en cuenta que en el mismo, el nucleótido o bien cambia su valor a otro diferente, o se mantiene igual su estado dependiendo del valor devuelto de Metropolis-Hasting. Concretamente, si este método retorna una base de nucleótido igual que aquel actualmente ocupado en el sitio dado, en este caso se dice que no hay mutación. De otro modo, el nucleótido ancestral se reemplaza por su descendiente diferente.

Capítulo 2. Implementación de los algoritmos para la simulación de las poblaciones virales

Hemos descrito en lo anterior 4 mecanismos de simulación, los cuales se pueden aplicar independientemente sobre la misma población de secuencias a fin de examinar el efecto de cada uno de ellos en el proceso evolutivo de las mismas. Observando que en una secuencia de ADN, el proceso de simulación se realiza en cada sitio, de forma tal que la secuencia generada tiene la misma longitud y se comporta como una mutación de su ancestro, formándose a partir de las componentes ya modificadas y evidentemente puede poseer propiedades biológicas diferentes.

Es necesario enfatizar que, con las mutaciones generadas de las secuencias ancestrales, hay que realizar un análisis de chequeo de la generación de los codones de terminación (codón de parada o stop codon) en cada secuencia obtenida. Como se había mencionado anteriormente en el capítulo 1, un codón de terminación es aquel codón (nucleótido triple) que no determina en el código genético aminoácido alguno. Su función es acotar el mensaje cifrado por el ADN que da lugar al ARN mensajero (mARN). En otras palabras, un codón de terminación señala a la maquinaria de la molécula que se ha alcanzado el fin del proceso de sintetizar las proteínas (Griffiths, 2000). En el código genético estándar, los codones de terminación vienen en tres diferentes formas: TGA, TAG y TAA en ADN y sus correspondientes copias UGA, UAG y UAA en ARN respectivamente.

El objetivo del chequeo de la generación de los codones de parada en las secuencias de mutación, es simplemente detectar el lugar donde aparecen los mismos a lo largo de las secuencias, luego es necesario quitar estos codones modificando las bases que forman parte de ellos. Este proceso se lleva a cabo de la siguiente forma:

1. Transformar la secuencia de nucleótidos codificados en el conjunto numérico {1, 2, 3, 4} correspondiente al conjunto alfabético {A, C, G, T}, a una matriz de dos dimensiones, en la cual cada columna contiene 3 elementos que se le corresponden a las 3 bases de cada codón. El número de columnas es el mismo de codones que tiene la secuencia.
2. Transformar esta matriz en un vector de números enteros de la base 4 que pertenecen al rango (1, 64), haciendo el intercambio de la primera y la segunda fila de la misma, y multiplicando cada columna de la matriz resultante con el vector (4^2, 4^1, 4^0). Al terminar este cálculo, obtendremos un vector de números enteros positivos que representan los codones de la secuencia. El tamaño de este vector es igual a la cantidad de codones.

3. Localizar los stop codones TAA, TAG y TGA (o UAA, UAG y UGA) en la secuencia mediante el vector entero anteriormente creado, conociendo que los valores representantes de estos codones son 33, 35 y 65, respectivamente.
4. Eliminar los stop codones de la secuencia, en caso que los mismos existieran, modificando las bases que forman parte de ellos de manera que la primera se mantiene igual, mientras que la segunda y la tercera se reemplazan por las otras bases diferentes generadas de forma aleatoria, empleando uno de los 4 procedimientos expuestos anteriormente. Este proceso se repite hasta que no quede ningún codón de parada en la secuencia.

La mutación de un ancestro determinado es aquella que resulta finalmente después de realizar este mecanismo de detección de los codones de parada. De esta forma, garantizamos que las secuencias generadas por el modelo evolutivo mantendrán cuantitativamente la completitud de la información genética de las mismas en el proceso de sintetizar las secuencias de amino ácidos, lo que se analizará posteriormente en las siguientes secciones.

Hasta aquí, hemos obtenido una descripción bastante detallada del proceso de simulación de la evolución molecular, aplicando la técnica de Monte Carlo. Hemos mostrado también que el algoritmo Metropolis-Hasting, conocido como el método más simple de esta técnica, la estimación del tiempo de espera evolutivo, y el uso de la matriz de transición, son los posibles procedimientos que pueden ser usados para generar la cadena de Markov, en la cual cada uno de los elementos representa un nuevo estado o, dicho en otra forma, una mutación de nucleótido, de una secuencia ADN a la otra, conduciendo así a la generación de una nueva secuencia diferente de la ancestral.

En la sección 2.3 se describirá el proceso de evolución de una población viral, en el cual la generación de las mutaciones juega un papel principal y es considerada como el paso inicial. Luego, éste es seguido por la selección de las secuencias que satisfagan algunas condiciones determinadas. Este proceso de selección no se lleva a cabo directamente sobre las mutaciones, sino a través de las secuencias de aminoácidos generadas mediante el proceso de síntesis de proteínas basado en el código genético. Para entender más claro de cómo se realiza la selección, realizaremos una explicación detallada sobre este procedimiento en la sección siguiente. Luego, en la otra, veremos cómo se implementan los algoritmos en el lenguaje de programación C++ para poner en práctica esta tarea.

2.2 Métodos de evaluar la exactitud de los modelos evolutivos en la reconstrucción de Q*

La estimación de matriz de tasas de sustitución *Q** es típicamente realizada con los escalares de tiempo arbitrarios, correspondiendo al hecho de que las escalas de tiempo absoluto no pueden ser estimadas por la observación de las secuencias desde un punto de vista de tiempo real. Por tanto, la estimación de *Q* usualmente se hace en una escala de tiempo diferente que la matriz de tasas *Q** asumida en la simulación (Keyuan X. 2005).

Con el objetivo de comparar las matrices *Q* y *Q**, se debe determinar un factor de escala ρ de forma tal que ρQ y *Q** estén en la misma escala de tiempo. En este trabajo, se utilizó una norma de Frobenius para determinar el factor escalar ρ apropiado. La norma de Frobenius de una matriz *A* está dada por la siguiente expresión:

$$\| A \|_F = \sqrt{\sum_i \sum_j a_{ij}^2} = \sqrt{tr(AA^T)} \tag{2.5}$$

El factor de escala óptimo se selecciona como una opción de ρ que minimice la cantidad $\| \rho Q - Q^* \|_F^2$. Es fácilmente determinado que la solución óptima de ρ se puede calcular mediante la siguiente fórmula:

$$\rho = \frac{\sum_{ij} q_{ij} q_{ij}^*}{\sum_{ij} q_{ij}^2} \tag{2.6}$$

Se usa el error relativo:

$$e = \frac{\| \rho Q - Q^* \|_F}{\| Q^* \|_F}, \tag{2.7}$$

como una medida para denotar la corrección de la matriz de tasas de sustitución estimada *Q*, con respecto a la matriz Q^* asumida en la simulación.

2.3 Descripción del proceso evolutivo de las poblaciones virales

Este proceso se simula partiendo de un conjunto de secuencias ADN correspondientes a los genes del virus de la influenza A/H1N1 captados en Internet. Cada uno de ellos es considerado como un ancestro, que pudiera generar un número de descendientes llamados mutaciones. La forma de seleccionar los primeros ancestros se lleva a cabo aleatoriamente, teniendo en cuenta

que la distribución de probabilidades de las secuencias en la población inicial se toma como una medida para realizar la selección, de modo que un ancestro se selecciona si el mismo tiene la distribución de probabilidad mayor que un número aleatorio generado. Es necesario señalar que el número de ancestros seleccionados puede ser fijo y, para cada uno de ellos la cantidad de descendientes a generar también pudiera ser determinada de manera tal que los diferentes ancestros tengan la misma cantidad de mutaciones.

La evolución de las poblaciones virales es un proceso bastante complejo que pudiera ser hecha bajo las restricciones evolutivas para realizar la selección de las secuencias que satisfagan cierta condición. En las secciones siguientes, veremos cómo se realiza la generación de las secuencias mutantes, en el caso de simulación sin restricciones y en el otro con la presencia de la presión selectiva del sistema inmune.

2.3.1 Generación de las mutaciones sin selección

En este caso, las secuencias mutantes se generan libremente para formar nuevas poblaciones a partir de los ancestros, sin chequear las condiciones evolutivas. Todas las secuencias generadas de esta manera se incluyen todas en la población del virus, y se clasifican según sean sus propiedades estructurales. Concretamente, este proceso se realiza mediante los pasos descritos a continuación:

- Paso 1: Especificar la cantidad de descendientes que pudiera tener cada ancestro.
- Paso 2: A partir de la secuencia ancestral seleccionada, generar una secuencia mutante aplicando la técnica de Markov.
- Paso 3: Clasificar la secuencia recién creada a través de la secuencia de amino ácidos sintetizada correspondiente. Si la misma coincide con uno de los alelos de la población, se aumentará el contador de éste en 1; de lo contrario, se agregará la mutante a la población como una nueva clase de virus. En los dos casos, la cantidad total de secuencias de la población se incrementa en 1.
- Paso 4: Calcular la distribución de probabilidades de la población actual.
- Paso 5: Tomar la secuencia mutante generada como el nuevo ancestro y volver el algoritmo al paso 2 si la cantidad de descendientes no se ha acabado; de lo contrario, moverse al paso siguiente.

- Paso 6: Seleccionar el nuevo ancestro entre las secuencias de la población actual. Si el número de ancestros no se ha terminado, el algoritmo vuelve al paso 1; de otro modo, trasladar al paso siguiente.
- Paso 7: Terminar el algoritmo.

Este procedimiento se realiza para cada generación del virus, y se repite tantas veces como el número de generaciones especificado al principio. Es importante enfatizar que una vez que se haya terminado de generar las mutaciones de una generación, es necesario actualizar la distribución de probabilidades de la población actual. De esta forma, los ancestros seleccionados en una generación podrían incluir a las secuencias que fueron creadas en las generaciones anteriores. Al finalizar este proceso, se obtiene una nueva población más amplia formada por varias generaciones que incluyen todas las secuencias mutantes, sin tener en cuenta ninguna condición evolutiva. En la sección siguiente, veremos cómo el proceso de generación de las mutaciones se simula cuando se aplica la presión selectiva sobre la población.

2.3.2 Generación de las mutaciones bajo la presión selectiva

Este proceso genera las secuencias de ADN mutantes mediante los mismos pasos descritos en el algoritmo anterior, excepto que entre el paso 2 y el 3 se aplica el mecanismo de selección. Este último consiste en el hecho de que, una nueva secuencia generada a partir de un ancestro dado, se analizará de forma tal que la misma pudiera satisfacer o no la condición de resistencia contra la vacuna. En el caso de que la mutación sea reconocida por la vacuna, entonces será eliminada definitivamente y, por supuesto, no sigue generando descendientes. De lo contrario, es decir, si la nueva secuencia no es reconocida por la vacuna, se agregará a la población, tal y como se ha descrito en el paso 3 anterior. La simulación computacional realiza la selección de las secuencias por la vacuna, mediante el análisis de regresión lineal. Este método estadístico se aplica en este caso de la forma siguiente:

- Transformar la secuencia de ADN generada a la secuencia de aminoácidos correspondiente, usando el código genético. Luego, convirtiéndose esta última en un vector de valores reales mediante las energías de los aminoácidos.
- Sobre este vector de energías, se crea un nuevo vector de medias, teniendo en cuenta el tamaño de la ventana de desplazamiento.

- Realizar el mismo proceso con la secuencia de vacuna para obtener otro vector de medias.
- Aplicar el modelo de regresión lineal simple sobre los vectores obtenidos, dado que la variable dependiente (denotada por *y*) corresponde a la media de la secuencia mutante, y la variable independiente (denotada por *x*) es correspondiente a la media de la secuencia vacuna.

Al encontrar el modelo de regresión adecuado, podemos determinar los residuos del mismo, y luego estudentizarlos para buscar los *outliers.* Estos últimos se usan para confirmar el hecho de que la mutación se reconociera o no por la vacuna. Si existen por lo menos 4 *outliers,* decimos que la vacuna no reconoce la secuencia mutante y ésta se agrega a la población; de lo contrario, la misma es eliminada.

Con todo lo que se ha explicado, podemos describir el algoritmo en los siguientes pasos:

- Paso 1: Especificar el número de descendientes que pudiera tener cada ancestro.
- Paso 2: A partir de la secuencia ancestral seleccionada, se genera una mutación aplicando la técnica de Markov.
- Paso 3: Transformar la secuencia ADN mutante y la vacuna a las secuencias de aminoácidos correspondientes, para luego convertir estas últimas en los vectores de energías y calcular los vectores de medias, conociendo el tamaño de la ventana de desplazamiento.
- Paso 4: Aplicar la técnica de regresión lineal para encontrar los residuos. Al estudentizar a estos últimos, se obtendrán los residuos que serán usados para encontrar los posibles *outliers*. La presencia de cuatro o más *outliers* es el criterio tomado para aceptar la nueva variante mutacional en la población. En el caso de ser aceptado, se continúa con el paso siguiente. De lo contrario, se elimina la mutante y se vuelve al paso 2.
- Paso 5: Clasificar la secuencia recién creada a través de la secuencia de amino ácidos sintetizada correspondiente. Si la misma coincide con uno de los alelos de la población, se aumentará el contador de éste en 1; de lo contrario, se agregará la mutante a la población como una nueva clase de virus. En los dos casos, la cantidad total de secuencias de la población se incrementa en 1.
- Paso 6: Calcular la distribución de probabilidades de la población actual.

- Paso 7: Tomar la secuencia mutante generada como el nuevo ancestro y volver el algoritmo al paso 2 si la cantidad de descendientes no se ha acabado; de lo contrario, moverse al paso siguiente.
- Paso 8: Seleccionar el nuevo ancestro entre las secuencias de la población actual. Si el número de ancestros no se ha terminado, el algoritmo vuelve al paso 1; de otro modo, trasladarse al paso siguiente.
- Paso 9: Terminar el algoritmo.

Este procedimiento describe la evolución de poblaciones virales con la aplicación de la restricción selectiva, de modo que no todas las secuencias mutantes generadas se agregan a la población, sino se escogen solamente las que satisfagan la condición de no reconocerse por la vacuna, o sea, las secuencias más resistentes. La operación de selección es un fenómeno común en la naturaleza y en la vida real, en que solo los individuos resistentes pudieran sobrevivir y adaptarse a los cambios frecuentes del ambiente.

Es evidente que el resultado obtenido mediante este proceso difiera del que no tiene en cuenta la restricción selectiva en el hecho de que el tiempo de ejecución crece y el tamaño de la población final es reducido. En la próxima sección, se expondrá la implementación de estos algoritmos con más detalle, y en el capítulo 3, analizaremos y discutiremos los resultados obtenidos mediante la aplicación de ambos algoritmos sobre una población de virus de la influenza.

2.4 Análisis de la complejidad temporal de los algoritmos

Para facilitar el cálculo de la complejidad temporal de los algoritmos, mostramos a continuación los pseudocódigos de los procedimientos y las rutinas principales que forman parte de la simulación.

Algoritmo: Simulación MCMC

Entradas:

s: Secuencia de nucleótidos de longitud n (n > 0)

m: Modelo de Markov

t: Vector de tasas de mutación por sitio

proc: Procedimiento de simulación usado

Salidas:

s': Secuencia mutante

p: Vector de probabilidades de mutación

prd : Vector de distribución de probabilidades

Pseudocódigo:

```
1.  Obtener la matriz de tasas ratesMat y el vector de frecuencias f del modelo m
2.  Para cada sitio i hacer
3.  |   Calcular la matriz de probabilidades probMat a partir de ratesMat
4.  |   Si (proc = 2) o ( proc = 3) hacer
5.  |   |   Calcular el tiempo de espera wt
6.  |   |   Si (wt < T) hacer
7.  |   |   |   Si (proc = 2) hacer
8.  |   |   |   |   Calcular matriz de transición transMat
9.  |   |   |   |   (s'(i), p(i), prd(i)) ← proc2(transMat)
10. |   |   |   Fin Si
11. |   |   |   Si (proc = 3) hacer
12. |   |   |   |   (s'(i), p(i), prd(i)) ← Roulette_Wheel(s(i), t, f, prd,
proc)
13. |   |   |   Fin Si
14. |   |   Fin Si
15. |   Si no   // proc = 1 o proc = 4
16. |   |   (s'(i), p(i), prd(i)) ← Roulette_Wheel(s(i), t, f, prd, proc)
17. |   Fin Si
18. Fin Para
```

19. *Chequear codones de parada en la secuencia mutante s'*
20. *Retornar s', p, pdr*

Algoritmo: Roulette_Wheel

Entradas:

b: Base de nucleótidos (b ∈ {1, 2, 3, 4})

p: Vector de tasas de mutación

f: Vector de frecuencia

pd: Vector de distribución de probabilidad

proc: Procedimiento de simulación usado

Salidas:

*b'**:** Nueva base de nucleótido*

d: Distribución de probabilidad

pd': Nuevo vector de distribución de probabilidades

Pseudocódigo:

```
1.  Normalizar el vector de tasas de mutación p
2.  Si (proc = 1) o (proc = 3) hacer
3.  |   Generar número aleatorio u
4.  |   Si (proc = 1) y (u < p[b]) hacer
5.  |   |       b' ← b
6.  |   |       d ← pd[b]
7.  |   |       pd' ← pd
8.  |   Si no
9.  |   |       Hacer
10. |   |   |       (b', d, pd') ← Metropolis-Hasting(b, p, f, pd)
11. |   |       Hasta (b' != b)
12. |   Fin Si
13. Si no       // proc = 4
14. |   (b', d, pd') ← Metropolis-Hasting(b, p, f, pd)
15. Fin Si
```

16. *Retornar b', d, pd'*

Algoritmo: Metropolis_Hasting

Entrada:

b: Base de nucleótido

p: Vector de probabilidades de mutación

f: Vector de frecuencia

pd: Vector de distribución de probabilidades

Salidas:

b': Nueva base de nucleótido

d: Distribución de probabilidad de la base

pd': Vector de distribución de probabilidades

Pseudocódigo:

```
1.  Generar número aleatorio u
2.  Calcular el vector de sumas acumulativas accSum de p
3.  Buscar i tal que accSum(i) > u
4.  b' ← i
5.  Si (b' != b) hacer
6.  |       r ← min(1, f[b]*p[b'] / (f[b']*p[b]))
7.  |       Generar número aleatorio v
8.  |       Si (r < v) hacer
9.  |       |       b' ← b
10. |       |       d ← pd[b]
11. |       |       pd' ← pd
12. |       Si no
13. |       |        d ← p[b]*r
14. |       |       Para i ← 1, mientras i<=4, inc(i)
15. |       |       |       rej[i] ← min(1, f[b]*p[i] / (f[i]*p[b]))
16. |       |       Fin Para
17. |       |       pd' ← (p*rej) / suma(p*rej)
```

```
18.   |       Fin Si
19.   Si no
20.   |       d ← pd[b]
21.   |       pd' ← pd
22.   Fin Si
23.   Retornar b', d, pd'
```

Algoritmo: stopCodonCheck

Entradas:

s: Secuencia de nucleótidos de longitud n (n > 0)

f: Vector de frecuencias

p: Vector de probabilidades de mutación entre nucleótidos

pm: Vector de probabilidades de mutación por sitio

prd: Vector de distribución de probabilidades

proc: Procedimiento de simulación usado

Salidas:

s': Secuencia mutante

pm': Nuevo vector de probabilidades de mutación por sitio

prd': Nuevo vector de distribución de probabilidades

Pseudocódigo:

```
1.    Transformar s a un vector entero intVec de codones codificados, tamaño n/3.
2.    Buscar los codones de parada en intVec y almacenar los índices en el vectoridx
3.    Si (tamaño(idx) != 0) hacer
4.    |       Para i←0, mientras i < tamaño(idx), inc(i)
5.    |       |       v ← 3*idx[i]-1
6.    |       |       (b, pr, vpr) ← Roulette_Wheel(v, p, f, prd, proc)
7.    |       |       s[v] ← b, pm[v] ← pr, prd[v] ← vpr
8.    |       |       v ← 3*idx[i]
9.    |       |       (b, pr, vpr) ← Roulette_Wheel(v, p, f, prd, proc)
10.   |       |       s[v] ← b, pm[v] ← pr, prd[v] ← vpr
```

11. | Fin Para

12. Fin Si

13. s' ← s; pm' ← pm; prd' ← prd

14. Retornar s', pm', prd'

Algoritmo: Evolution_Model

Entradas:

V: Vector de n secuencias de ADN

Freq: Vector de frecuencias

m: Modelo de Markov

sitesRate: Vector de tasas de mutación por sitio

proc: Procedimiento de simulación usado

Salidas:

V': Vector de secuencias mutantes

Freq': Vector de frecuencias actual

Pseudocódigo:

1. *Leer secuencias desde fichero y almacenarlas en V*
2. *Leer frecuencias desde fichero y almacenarlas en Freq*
3. *Seleccionar número de generaciones (ng), número de ancestros (na), y número de descendientes (nd)*
4. *Para i ← 1 hasta ng hacer*
5. | *Para j ← 1 hasta na hacer*
6. | | *Sleccionar un ancestro ances*
7. | | *Para k ← 1 hasta nd hacer*
8. | | | *(s, p, prd) ← MCMC(V[ances], m, sitesRate, proc)*
9. | | | *Clasificar s*
10. | | | *Insertar s en V*
11. | | *Fin Para*
12. | | *Actualizar el vector frecuencias Freq*
13. | *Fin Para*

14. *Fin Para*
15. *Retornar V' ← V, Freq' ← Freq*

Analizamos la complejidad de los algoritmos partiendo desde el más elemental, la rutina ***Metropolis-Hasting***. En este algoritmo, las instrucciones 1 a 4 realizan cálculos simples incluyendo la generación del número aleatorio (instrucción 1), el cálculo de la suma acumulativa de un vector de 4 elementos (instrucción 2), la búsqueda de un valor dado sobre el mismo vector y la asignación (instrucciones 3 y 4), todas ellas tienen costo $O(1)$. Las instrucciones de 5 a 22 son condicionales en las que la evaluación de las condiciones son también $O(1)$, las demás sentencias son de cálculos sencillos y asignaciones, excepto las instrucciones 14 a 16 que realizan un ciclo. Sin embargo este ciclo sólo repite 4 veces los cálculos y asignaciones simples, por tanto también tienen costo $O(1)$. De todos los análisis anteriores, se concluye que la rutina ***Metropolis-Hasting*** tiene un costo total de $O(1)$.

El algoritmo ***Roulette_Wheel*** realiza la primera instrucción para normalizar un vector del tamaño 4, el costo de la misma es *O(1)*. El resto de las instrucciones son sentencias condicionales, las cuales incluyen los cálculos numéricos y asignaciones. En las instrucciones 10 y 14 se hacen las llamadas del algoritmo Metropolis-Hasting, el cual, como se ha analizado anteriormente, tiene la complejidad de *O(1).* Por tanto, el costo total de ***Roulette_Wheel*** sigue siendo *O(1).*

En el caso de la rutina ***stopCodonCheck***, la instrucción 1 realiza la transformación de la secuencia de nucleótidos de longitud n a un vector de valores enteros del tamaño $n/3$, luego la instrucción 2 hace una búsqueda en el mismo vector los codones de parada. Ambas instrucciones tienen costo *O(n)*. A continuación, las instrucciones de 3 a 12 son condicionales, donde se realiza un ciclo que pudiera, en el peor caso, repetir n veces un conjunto de los cálculos simples y asignaciones, incluyendo la función *Roulette_Wheel()*, cada una de las cuales tiene costo *O(1)*. Por tanto, la complejidad total de aquellas instrucciones es *O(n).* Finalmente, por "*la ley de la Suma",* se concluye que la rutina ***stopCodonCheck*** tiene complejidad total de $O(n) + O(n) + O(n) = O(max\{n, n, n\}) = O(n)$.

El algoritmo ***MCMC*** empieza con la obtención de la matriz de tasas de sustitución y el vector de frecuencias de nucleótidos a partir del modelo markoviano usado, la complejidad es $O(1)$.

A continuación, el algoritmo entra a un ciclo que repite *n* veces, tanto como la longitud de la secuencia de nucleótidos. Las instrucciones que se ejecutan en este ciclo, de 2 a 18, incluyen las operaciones de asignación, los cálculos numéricos y las llamadas a la función *Roulette_Wheel()*, las cuales tienen el mismo costo de *O(*1*)*. Por tanto, la complejidad de este ciclo es *O(n)*. Por último, la instrucción 19 llama a la función *stopCodonCheck*, la cual, como se ha analizado anteriormente, tiene complejidad *O*(*n*), siendo siempre *n* la longitud de las secuencias genómicas de ADN. Finalmente, podemos concluir que la complejidad total del algoritmo ***MCMC*** es *O(n) + O(n) = O(max {n, n}) = O(n).*

El algoritmo ***Evolution_Model se encarga de ejecutar el proceso completo de simulación,*** en el cual las tres primeras instrucciones realizan la lectura de datos desde los ficheros de entrada, las cuales tienen complejidad *O*(1). Luego, el algoritmo se continúa con 3 ciclos anidados, el más profundo incluye el método *MCMC* que tiene costo *O*(*n*); ambas funciones de clasificación e inserción de la secuencia en la población también son de complejidad *O*(*n*). Por tanto, el ciclo más interno tiene costo $n*[max\{O(n), O(n)\}] = n*O(n) = O(n^2)$. El segundo ciclo incluye además una sentencia de selección de ancestro y la otra de actualización del vector de frecuencias, la primera de ellas tiene costo *O*(1), mientras que la segunda es de O(n), lo que conduce a la complejidad $O(n^3)$ de este ciclo. Finalmente, el ciclo más externo tiene complejidad $O(n^4)$, éste es también el costo total del algoritmo ***Evolution_Model.***

En conclusión, la combinación de todos los algoritmos definidos anteriormente tiene complejidad $O(n^4)$.

2.5 Implementación de los algoritmos de simulación usando el paradigma de programación orientada a objetos

En esta sección se presenta una descripción del diseño general de la aplicación, en el cual los algoritmos presentados en las secciones anteriores forman partes del procedimiento de la simulación. Se introducen, además, los diagramas de clases que representan las entidades construidas en la aplicación y relaciones entre ellas, en ambos casos de simulación, a fin de darle al lector una vista general y una mejor comprensión del desarrollo completo de la aplicación con respecto al paradigma de programación orientada al objeto.

2.5.1 Diseño general

Este trabajo desarrolla una aplicación que toma como entradas un alineamiento de secuencias de ADN y un vector de frecuencias de distribución correspondientes a las mismas, a partir de las cuales, se desea aplicar el algoritmo de simulación teniendo en cuenta primero la especificación de un modelo de Markov que se usará, los parámetros del modelo seleccionado, y el vector de las tasas de sustitución por sitio. Para esto último, el software brinda un menú de los modelos disponibles para que el usuario escoja el que desea usar, y dos opciones para obtener los parámetros del modelo y el vector de tasas, los cuales, o bien, se estiman a partir del alineamiento, o se introducen directamente desde ficheros. Una vez que se hayan obtenido los datos anteriores, es necesario especificar los parámetros de la simulación, los cuales incluyen el tiempo evolutivo y el procedimiento de la simulación usado.

Las secuencias mutantes generadas pudieran ser pasadas luego a realizar el proceso de selección en el caso que la simulación sea procesada bajo la condición selectiva y, finalmente, las mismas se clasifican según sea el tipo de alelo al cual pertenezcan. Todo el proceso de la simulación se muestra en el diagrama de la ***Figura 2.1***.

En las dos secciones siguientes de este capítulo, presentaremos los esquemas del diseño para la implementación de los algoritmos de simulación, tanto en el caso sin restricción como en el caso en que se aplica la condición selectiva. Se explicará con detalle la forma de construcción del sistema completo y el uso de las herramientas computacionales necesarias en la implementación de los algoritmos para ahorrar el tiempo de programación y lograr la eficiencia del trabajo.

2.5.2 Simulación de las poblaciones en el caso sin restricción

La aplicación se ha construido como un módulo compuesto de varios sub-módulos, basándose en la técnica de programación orientada a objetos. Cada sub-módulo se presenta como una clase que se caracteriza por los atributos y operaciones, y se encarga de realizar una tarea determinada, teniendo en cuenta la relación con respecto a otras clases que forman parte del modelo.

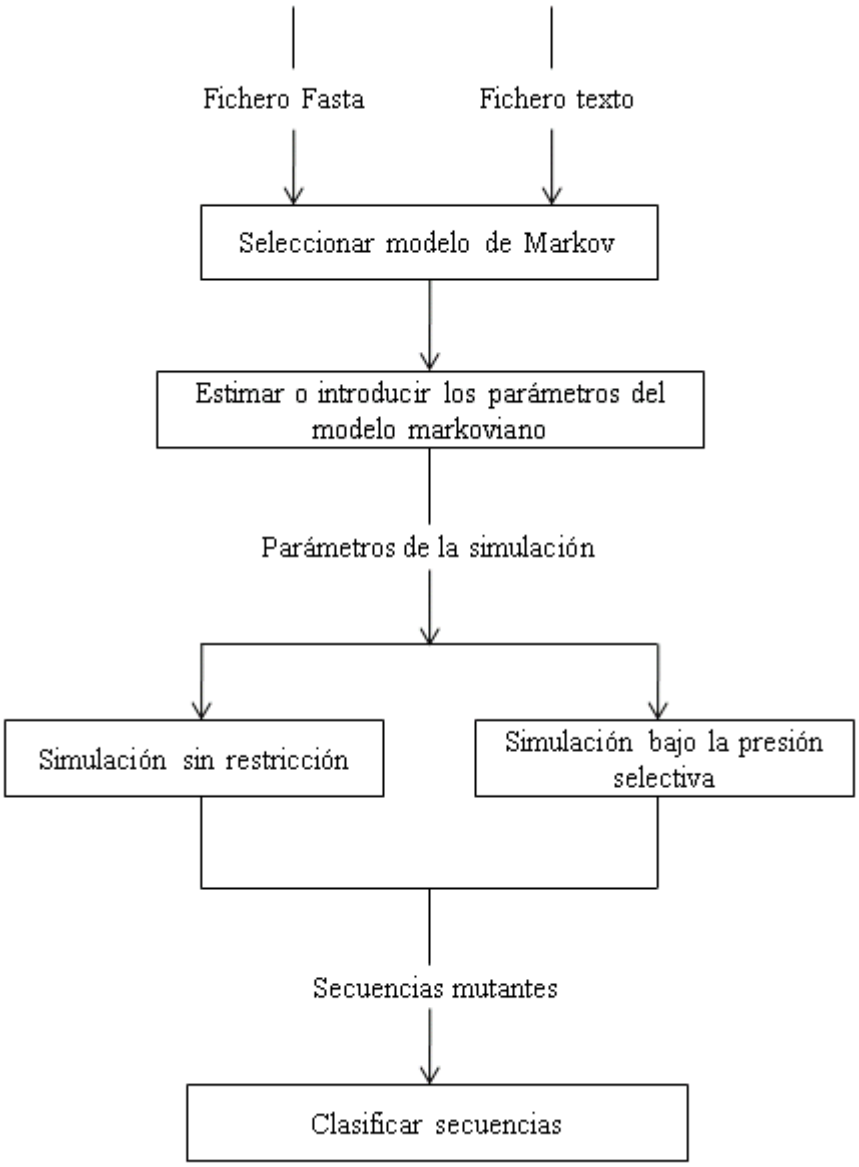


Figura 2.1. *Diagrama general del proceso de simulación de poblaciones virales.*

Como se ha dicho anteriormente, la simulación de la evolución de las poblaciones virales pudiera ser hecha bajo restricciones evolutivas para generar solo las secuencias que satisfagan cierta condición, o pudiera ser realizada de manera que las mismas se generan libremente sin limitación alguna. En el diagrama de la ***Figura 2.2***, se muestra el modelo de simulación en el caso sin restricciones, que muestra las clases utilizadas y las relaciones existentes entre ellas. Observando el esquema, se puede ver que el sistema se constituye por 3 bloques principales:

1. El primer bloque está compuesto por las clases que son consideradas como básicas del modelo que se encarga de realizar las tareas relacionadas con el procesamiento de las secuencias genómicas, refiriéndose a la clase *DNA_Sequence*. Además, con el mismo propósito, se utilizan las clases *Matrix* y *Vector* de la librería estándar del lenguaje C++ ya mencionada antes y nombrada BTL (Biomodeling Library for C++), para facilitar los cálculos y operaciones numéricos complicados.
2. El segundo bloque consiste en la resolución de la tarea principal del problema, lo cual se compone por 5 clases: *Roulette, MC_MonteCarlo, Sequences_Algorithm, Param_Estimator* y *Evolution_Model*. La primera clase que debe ser usada en la simulación es *Param_Estimator,* que realiza la estimación de los parámetros del modelo de Markov seleccionado y, a partir de los mismos, calcula el vector de tasas de mutación por sitio de las secuencias. Los métodos implementados de esta clase se pueden invocar cuando empiece la ejecución del programa, con el objetivo de obtener los datos necesarios que serán usados como datos de entrada para otros algoritmos de la simulación. Entre otras clases de este bloque, *Roulette* es la más sencilla, y se encarga de realizar la transición de nucleótido (A, C, G o T) en un sitio dado de la secuencia, teniendo en cuenta el estado actual, la tasa de sustitución y la distribución de probabilidad del nucleótido ocupado en ese sitio. De hecho, la simulación del proceso evolutivo de una población de secuencias genómicas se lleva a cabo mediante la aplicación del método *Roulette* para cada secuencia, con el objetivo de generar luego una nueva población de secuencias mutantes a partir de un conjunto de secuencias ancestrales, de lo cual se encarga la clase *MC_MonteCarlo.* Estas mutaciones se aceptarán como los nuevos individuos de la población, una vez que se haya chequeado la existencia de los codones de parada en las mismas, para reemplazarlos por otros codones generados de forma aleatoria mediante el método *stopCodonCheck()* de la clase *Sequences_Algorithm*. Por último, la clase *Evolution_Model* es la más importante que simula el proceso evolutivo de una población viral, y en ella se hacen las llamadas al método MCMC para generar las secuencias mutantes, dado el modelo evolutivo de Markov y la especificación de los parámetros de la simulación.

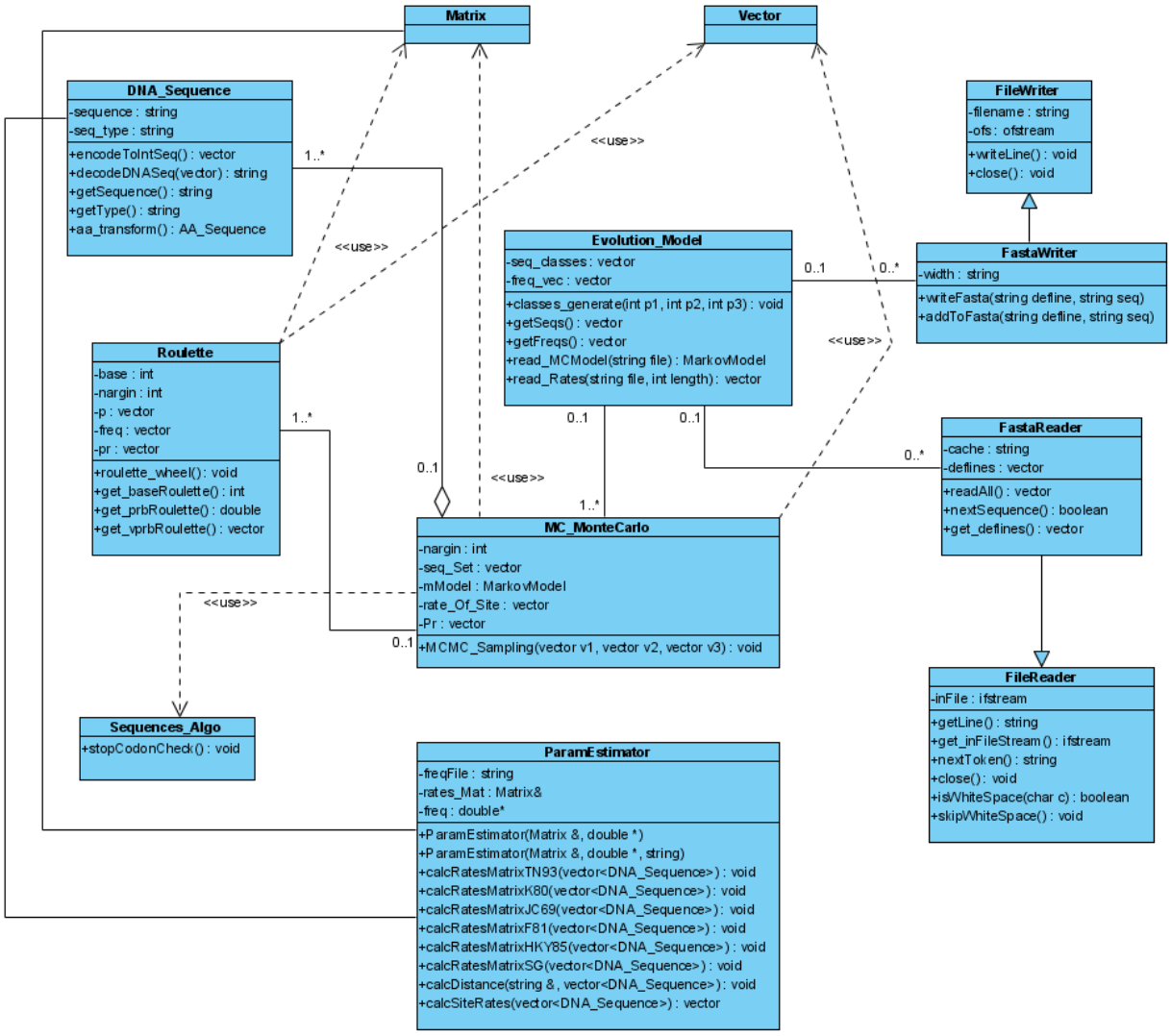


Figura 2.2. *Diagrama de clases para la simulación sin restricción*

3. El tercer bloque del sistema es responsable de realizar los procesos de leer y escribir datos desde y hacia los ficheros. Para ello, se han usado las clases *FastaReader* y *FastaWriter* que heredan de las clases superiores *FileReader* y *FileWriter*, respectivamente. La primera es usada para la lectura, mientras que la segunda es para la escritura. Ambas clases trabajan con los ficheros del formato Fasta, los cuales se caracterizan por la estructura de almacenamiento de las secuencias de ADN, especificando un encabezamiento identificador de la secuencia, y luego la secuencia alineada. La herencia de las clases superiores es útil en el caso de realizar tareas simples con los ficheros como,

por ejemplo, el chequeo de espacios en blanco, recuperación de las líneas, cerrar ficheros, etc. Las clases *FastaReader* y *FastaWriter* forman parte de la librería BOOM (Bioinformatics Object-Oriented Modules).

2.5.3 Simulación de las poblaciones bajo la presión selectiva

El procedimiento de selección de las secuencias mutantes, que satisfagan cierta condición evolutiva, se introduce al modelo de simulación cuando es necesario chequear la resistencia de las secuencias genómicas capaces de evadir la acción del sistema inmune, inducida por la vacunación. Para definir la secuencia de ADN viral como una vacuna, se escoge, entre otras variantes mutacionales, aquella que al ser comparada con el ancestro en cuanto a las características físico-químicas, mediante el análisis de regresión lineal, no presenten más de cuatro *outliers*. La misma es usada para seleccionar, entre las secuencias descendientes, aquellas que son probabilísticamente *resistentes a la vacuna*, de modo que las mismas se añaden como nuevos individuos a la población del virus de influenza.

El esquema de la ***Figura 2.3*** muestra el diseño de clases para el modelo de simulación bajo la presión selectiva de las secuencias mutantes. Al igual que en el caso presentado anteriormente, este modelo se construye mediante el diseño estructural de algoritmos dividido en tres bloques de diferentes funcionamientos, excepto que en el primer bloque definimos una nueva clase nombrada *AA_Sequence*, la cual se encarga de las tareas relacionadas con las secuencias de aminoácidos. Esta clase se utiliza de forma asociativa con la clase *DNA_Sequence*, con el objetivo de transformar esta última a la secuencia de aminoácidos correspondiente. La transformación es útil para el proceso de selección pues la misma se ejecuta, como el primer paso, en el análisis de regresión lineal entre la secuencia viral mutante y la secuencia de la vacuna. Las demás operaciones necesarias, que se aplican sobre las secuencias de aminoácidos en el análisis de regresión, se definen también en la clase *AA_Sequence* y se utilizan como pasos iniciales de este procedimiento.

Otra modificación que se ha realizado al modelo de simulación es la introducción de tres nuevas clases en el segundo bloque para hacer el análisis de regresión lineal, que se denominan *LinearFunc*, *LinRegresor* y *Regresión*. Las dos primeras clases vienen de la librería BOOM mencionada anteriormente, mientras que la tercera fue implementada por la

autora de esta tesis para calcular los residuos de la regresión y realizar el proceso de estudentizarlos, con el fin de encontrar luego los *outliers* del modelo.

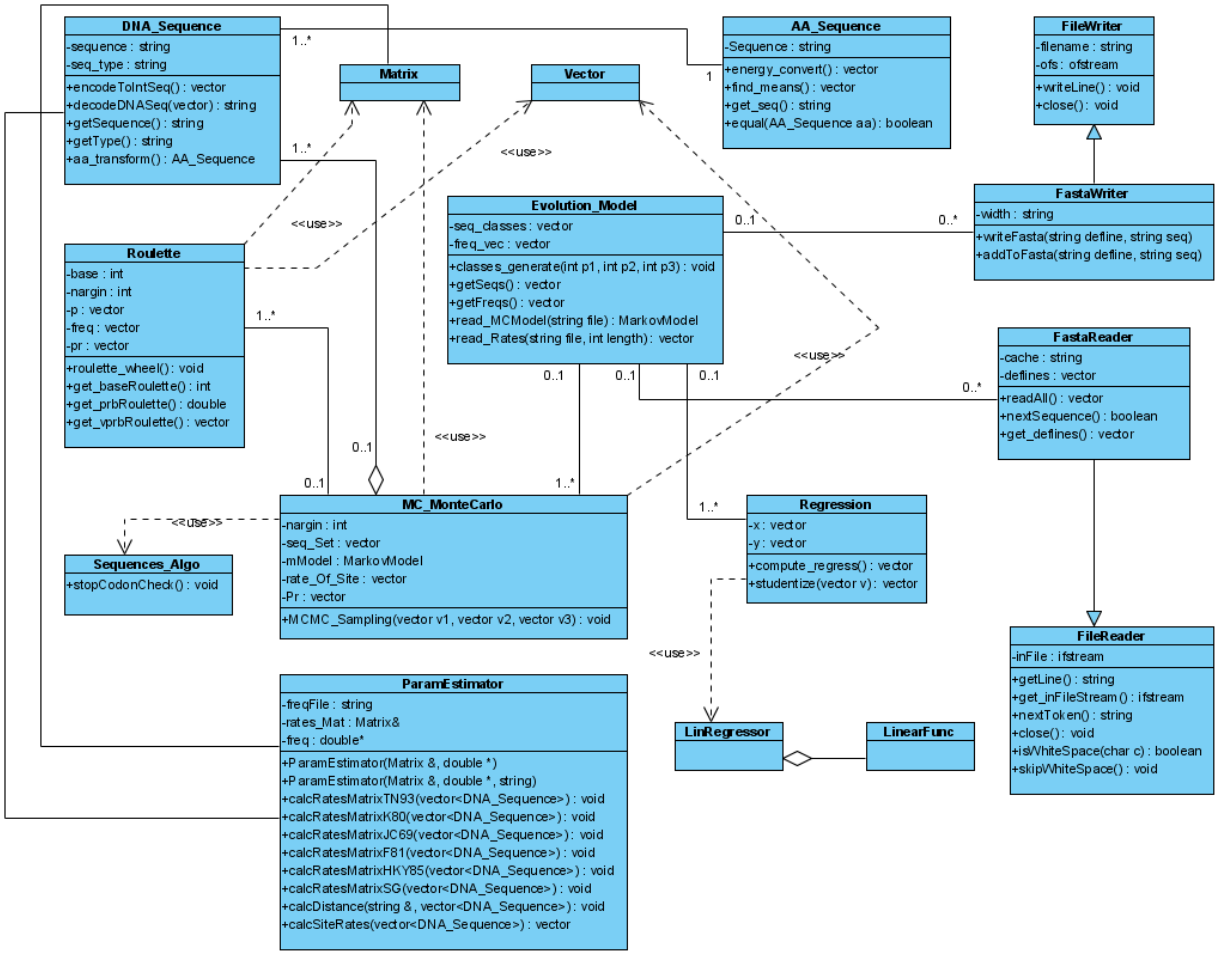


Figura 2.3: *Diagrama de clases para la simulación bajo la presión selectiva*

Mediante los diagramas de clases, es fácil ver que el corazón del algoritmo de simulación se encuentra en el segundo bloque, donde se aplica el método clásico de cadenas de Markov para generar, a partir de una población inicial de secuencias ancestrales, una nueva población más amplia de mutantes, de forma tal que el proceso de simulación se realiza distribuyendo las tareas, entre diferentes sub-modelos, desde la más simple hasta la más compleja, para obtener en una sola clase el resultado final de la simulación.

Esta forma de dividir trabajos, basada en los conceptos de clases, solamente la podría tener un lenguaje de programación orientado a objetos. El mejor y más poderoso de los lenguajes que

hemos usado es C++, que brinda una forma muy eficiente en la implementación, la compilación, así como el uso de librerías (bibliotecas) estándares disponibles del lenguaje, como las ya mencionadas. Otra ventaja que podemos obtener, al trabajar con clases y objetos, es la facilidad de modificar el sistema cuando es deseable cambiar algún método o alguna variable de una clase, de forma tal que este cambio no influya a los métodos de otras clases. Esta ventaja, conocida como la propiedad de encapsulamiento del lenguaje orientado a objetos, es una de las razones apropiadas por la cual se selecciona C++ como la mejor herramienta de programación en este trabajo, con el objetivo de producir un software de simulación eficiente, además de eficaz. No se puede obviar que estas experimentaciones requieren de un volumen considerable de tiempo, que es necesario acotar para lograrla en un período razonable y el C++ ayuda a esto.

2.6 Conclusiones parciales del capítulo

La aplicación se ha desarrollado como una herramienta simuladora de secuencias genómicas basada en la técnica de Monte Carlo, con el objetivo de simular el proceso evolutivo de las poblaciones virales, tanto en el caso sin restricción como en el caso en que se aplica la presión selectiva del sistema inmune, esto último con el fin de detectar las variantes mutacionales de escape. Por la necesidad de tener un sistema integrado de simulación capaz de aplicarse eficientemente sobre cualquier base de datos, se ha introducido al software la implementación de varios modelos evolutivos y diferentes mecanismos de simulación basados en el método de cadenas de Markov. Se incluyen también, en el mismo sistema, los métodos estadísticos para evaluar la exactitud de los modelos markovianos en la reconstrucción de matriz de tasas de mutación. Se presentó un análisis de la complejidad temporal de los algoritmos que se implementan sobre el paradigma de programación orientada a objetos usando el lenguaje C++ para construir el sistema. Se expuso también el diseño general de la aplicación, en el cual se puede ver claramente el flujo completo de todo el proceso de simulación y, finalmente, los diagramas de clases que forman parte del diseño preliminar del software.

3 RESULTADOS Y DISCUSIÓN

En el presente capítulo se exponen los resultados del simulador desarrollado al aplicarse el mismo sobre una población viral de la influenza A/H1N1. En particular, se describen las distintas pruebas realizadas tomando diferentes valores de los parámetros de la simulación. Luego se hace un análisis de los resultados obtenidos, mediante el diseño experimental, para descubrir la mejor condición evolutiva, sobre la cual se obtiene una buena predicción de nuevas variantes mutacionales capaces de evadir la presión selectiva del sistema inmune.

3.1 Descripción de la base de datos

Se utiliza en este trabajo un alineamiento de 65 secuencias genómicas de ADN codificantes para la proteína Hemaglutinina (HA), una lecitina que es el sitio antigénico más importante del virus de influenza A/H1N1 porque las mutaciones en este sitio reducen o inhiben la unión de anticuerpos neutralizantes, formándose un nuevo subtipo. Este fenómeno es conocido como «drift antigénico», las mutaciones que se desarrollan por efecto de este fenómeno son la explicación de las epidemias estacionales en climas fríos durante el invierno, por ejemplo en Cuba (Sánchez R. and Grau R. 2009).

Estas secuencias fueron aisladas en el periodo desde enero hasta junio de 2009 y tomadas de la base de datos del GenBank en el NCBI (http://www.ncbi.nlm.nih.gov/), cuya descripción se muestra en la tabla A1 del Anexo 1. Las mismas secuencias forman la población inicial y se consideran como diferentes individuos, los cuales son identificados por el nombre de la especie del que fue aislado, el país de origen, la fecha en que se produjo el aislamiento y el código de acceso a la base de datos. La población tiene una distribución no uniforme de manera tal que los individuos tienen diferentes frecuencias.

El alineamiento múltiple fue obtenido con el MEGA 4, ignorando los gaps. La razón por la cual se toman las secuencias de ADN sin gaps es por el hecho de que la simulación pretende predecir las secuencias mutantes, que pudieran ser generadas en condición de presencia o ausencia de la presión selectiva, de modo que las secuencias de ADN codificadas a las proteínas correspondientes se comparan con las secuencias de proteínas existentes de las bases de datos reales, en cuanto a la puntuación total del alineamiento y el por ciento total de

identidad. La presencia de los gaps en el alineamiento de las secuencias no es necesaria, e incluso, pudiera tener el efecto de producir ruido en la comparación, lo cual influye notablemente en la exactitud de los resultados.

3.2 Aplicación de la simulación sobre una población de virus real

La mejor forma de examinar el funcionamiento de un software es ponerlo en práctica, mediante la realización de diversas pruebas experimentales. En este trabajo, debido a que la simulación propone para cada prueba un gran número de iteraciones, a fin de generar nuevas poblaciones de secuencias genómicas, es de importancia tener en cuenta el rendimiento de la aplicación, en cuanto al tiempo de ejecución computacional. Para ello, fue recomendable realizar las pruebas sobre la plataforma del sistema operativo Linux, pues el mismo siempre favorece mejor el lograr la eficiencia de la aplicación en comparación con el sistema Windows.

En el Anexo 2, se muestran los gráficos de los resultados obtenidos, tanto para el caso de simulación sin restricciones como para el caso bajo la presión selectiva. En cada prueba, es necesario especificar valores para los parámetros de la simulación, los cuales incluyen el tiempo evolutivo (t), que varía su valor en el intervalo [0.02, 0.06], uno de los cuatro procedimientos de simulación *(proc)* y el modelo de Markov *(m)* deseado. Asumiendo que en todas las pruebas, la población inicial evoluciona a través de 7 generaciones, por cada generación se seleccionan 7 ancestros y cada uno de ellos genera 5000 descendientes. Obsérvese que en cada figura, se presentan los gráficos de las pruebas realizadas en ambos casos de simulación, las cuales comparten los mismos valores de los parámetros. Esta forma de representación facilita la comparación par a par entre los dos casos, en cuanto al número de alelos (variantes de una secuencia) generados y a la frecuencia relativa de cada uno de ellos en la nueva población. Debido a que la simulación sin restricciones genera secuencias mutantes y las introduce libremente a la población, mientras que la simulación bajo la presión selectiva impone restricción sobre las mutantes, de forma tal que solo se aceptan aquellas que satisfagan la condición establecida, el tamaño de esta última población se reduce.

Para cada prueba experimental, el software ofrece al usuario la opción de especificar, entre otros parámetros de simulación, el modelo evolutivo de Markov, que puede ser uno de los 6

modelos implementados. Con el propósito de obtener el mejor resultado de predicción, las valoraciones experimentales fueron realizadas tomando todas las combinaciones posibles de los parámetros. De hecho, es posible determinar los valores apropiados de dichos parámetros que mejor predicen las variantes mutacionales y, por tanto, se obtiene el modelo markoviano adecuado para la simulación.

Es importante señalar también que el valor predictivo de las simulaciones realizadas está dado por la capacidad de generar secuencias mutantes, a partir de la misma población inicial, que se asemejen a las que efectivamente se observan en la naturaleza. Las frecuencias relativas de los alelos generados se toman como una medida para seleccionar aquellos que mantienen una frecuencia elevada en las generaciones sucesivas. Luego se realiza la búsqueda, sobre los mismos, usando el algoritmo BLAST del MEGA 4, para encontrar las secuencias naturales existentes en la base de datos GenBank del NCBI que se asemejen a las generadas. La similitud está dada por la puntuación del alineamiento (*Alignment Score*), la cual se considera como el criterio de comparación entre las secuencias generadas por la simulación y las observadas en la realidad.

En los anexos 4 y 6 se muestran, para cada caso de combinación de los parámetros de simulación, los valores medios de las puntuaciones del alineamiento de los alelos con mayor frecuencia en la población generada, así como los porcientos de similitud correspondientes entre las secuencias generadas y las observadas en la base de datos Influenza Virus Resources. Obsérvese que diferentes combinaciones influyen notablemente en el valor predictivo de las simulaciones. A pesar de que producto del tiempo computacional, la cantidad de las pruebas experimentales realizadas no es muy elevada, fue posible determinar la combinación apropiada de los parámetros que constituye la mejor simulación, mediante el método de diseño experimental llamado *Metodología de Superficie de Respuesta* (Douglas 2005). Este método será presentado y discutido en la sección 3.4 de este capítulo.

3.3 Comparación entre dos casos de simulación en cuanto al número de alelos producidos

Obviamente, los resultados obtenidos en las pruebas experimentales dependen de la asignación de valores a los parámetros de simulación. Además, en condición de ausencia y bajo la presión

selectiva, las pruebas realizadas producen marcadamente diferentes cantidades de alelos. Este resultado es de esperar, pues, como se había dicho previamente, la evolución de las poblaciones virales se ve afectada por la presión selectiva del sistema inmune. Concretamente, a las secuencias mutantes, una vez generadas, se les evalúa la consistencia a la acción inducida por la vacuna, de forma tal que sólo aquellas no son detectadas por la misma, se aceptarán como nuevos alelos a la población, de lo contrario serán eliminadas. Por lo tanto, el tamaño de las poblaciones obtenidas bajo la selección resultaría más pequeño y la cantidad de alelos generados sería menor, en comparación con el caso de simulación sin restricciones. La tabla A3 del Anexo 3 muestra las cantidades de alelos producidos en las pruebas experimentales en dos casos de simulación. Teniendo en cuenta que la comparación se realiza entre las secuencias previamente codificadas a proteínas, todas aquellas que codifican para la misma proteína se agrupan y se identifican por un solo valor de alelo, observando que existe una diferencia clara en el número de alelos entre los dos casos. Específicamente, el tamaño de la población resultante en el caso sin selección siempre es mayor que el de la población generada en el caso de presión selectiva, conociendo que en ambos casos se comparten las mismas combinaciones de los parámetros.

Este resultado confirma una vez más, el hecho de que la evolución de las poblaciones virales está restringida por la presencia de la presión selectiva del sistema inmune, inducida por la vacunación, de modo que solamente las secuencias no detectadas por la vacuna se introducen a la población y continúan evolucionando, mientras que aquellas que son detectadas, se eliminan definitivamente, lo cual conduce a la reducción del tamaño de las poblaciones generadas.

En las últimas secciones de este capítulo, se discutirá el método del diseño experimental usado para modelar la simulación del proceso evolutivo de poblaciones virales. Luego se realizará un breve análisis de los resultados obtenidos mediante la aplicación de este método para los dos casos de simulación y, finalmente, se presentarán las conclusiones fundamentales del capítulo.

3.4 Diseños experimentales basados en el método de superficie de respuesta (RSM)

La Metodología de Superficie de Respuesta, o RSM por sus siglas en inglés, es un conjunto de técnicas matemáticas y estadísticas utilizadas para modelar y analizar problemas en los que una variable de interés es influenciada por otras. El objetivo es optimizar la variable de interés. En todos los problemas de RSM, la relación entre la respuesta y las variables independientes es desconocida, por lo tanto el primer paso en RSM es encontrar una aproximación adecuada para la relación funcional entre la variable respuesta y el conjunto de las variables independientes (Douglas 2005). A esta relación se le llama *Función de Respuesta predicha*, la cual se puede presentar con una ecuación polinomial. El éxito en una investigación de superficies de respuesta depende de que ésta se pueda ajustar modestamente a un polinomio de primer o segundo grado. Para tener un conocimiento más profundo de cómo se sigue este procedimiento hasta encontrar un óptimo y los diseños experimentales que se pueden utilizar, el lector puede consultar el libro titulado "*Response surface Methods from Design and Analysis of Experiments*" de Douglas C. referenciado anteriormente.

En las secciones siguientes, presentaremos los diseños experimentales para ambos casos de simulación, conociendo que la variable de respuesta es la puntuación del alineamiento de las secuencias genómicas de ADN (*Alignment Score*), mientras que las variables independientes son dos parámetros de la simulación, los cuales incluyen el tiempo evolutivo (t) y el procedimiento de simulación *(proc)*. El objetivo es encontrar un modelo adecuado que ajuste la relación entre las variables, el cual puede determinar la mejor combinación de los parámetros que optimice el valor de la variable respuesta. En todos los casos se empleó un diseño experimental de superficie de repuesta centrado. En este diseño se realiza la replicación de las condiciones centrales del experimento a fin de estimar el error experimental y evaluar la calidad del modelo.

La región de exploración para ajustar los modelos se encuentra limitada en un espacio bidimensional, donde el tiempo evolutivo (t) está ubicado en el intervalo [0.02, 0.06] y el procedimiento de simulación (*proc*) puede ser uno de los cuatro definidos. Para simplificar los cálculos, las variables independientes se codifican de la forma siguiente:

$$x_1 = \frac{t - 0.04}{0.02}, \qquad x_2 = \frac{proc - 3}{1} \tag{3.1}$$

Donde los valores 1 y 0.02 son definidos como pasos entre los puntos de intervalo, mientras que las variables x_1, x_2 son codificaciones de los parámetros t y *proc*, respectivamente, y pueden tomar valores enteros en la región [-2, 1]. Los diseños experimentales, para la simulación sin restricciones y la simulación bajo la presión selectiva, se muestran en las tablas de los anexos 4 y 6, respectivamente, donde la última columna de las mismas representa el valor medio de puntuación de alineamiento de las secuencias mutacionales generadas en cada prueba experimental, que tiene la mayor frecuencia relativa en la población. Nótese que los diseños experimentales se realizan abordando todos los valores de las variables independientes sobre su espacio de búsqueda, y además se extiende por la realización de 5 réplicas del experimento correspondiente al punto central, es decir, el que se corresponde a la combinación de los parámetros: $t = 0.04$, $proc = 3$.

En ambos casos de simulación no usamos el parámetro *m*, que denota al modelo evolutivo de Markov, como una variable independiente para los diseños experimentales. Esto es debido al deseo de reducir el espacio de búsqueda para que el mismo sea bidimensional. Este parámetro es utilizado para dividir el conjunto de las pruebas experimentales en subconjuntos diferentes, de modo que cada uno se corresponda a un modelo de Markov usado para la simulación. Esta forma de clasificar las pruebas tiene el propósito de examinar el valor predictivo de simulación de cada modelo y, de hecho, poder descubrir el modelo, entre otros, que mejor prediga las variantes mutacionales de las poblaciones virales.

Una de las herramientas computacionales que se puede usar para aplicar los métodos de diseños experimentales es el paquete estadístico, que también es un lenguaje de programación, llamado *R* (Brian S. 2009), en el cual se dispone una variedad de las funciones de la librería nombrada *rsm* que son útiles para diseñar y analizar los experimentos hechos secuencialmente, esperando lograr una superficie de respuesta óptima. Se recomienda que el lector vea el libro *Response-surface Analysis* (Lenth 2009) o el artículo *Response-Surface Methods in R, Using rsm* (Lenth 2009) para tener una mejor comprensión y ser capaz de utilizar los métodos de *rsm* en *R*.

3.4.1 Diseños experimentales para la simulación de poblaciones virales en ausencia de la presión selectiva

Al aplicar el método de ajuste del modelo de segundo orden conocido como SO, las salidas correspondientes a diferentes modelos evolutivos de Markov se obtienen tal y como se muestran en las tablas del Anexo 5. En cada uno de ellas, el análisis de ajuste se presenta mediante la tabla de análisis de varianza de regresión, donde se muestran las diferentes fuentes de variación que contribuyen a la variación total de los datos. A esta última se le llama *La suma de cuadrados total* (*Sum Sq*). Se plantean también en el mismo análisis las pruebas de significación para el cociente de la suma de cuadrados explicada de la regresión (explicada por cada uno de los términos del modelo) y la suma de cuadrado de los residuales. La significación se estima utilizando el estadístico F de Fisher. En particular, en el modelo SO se plantea una prueba de hipótesis para cada conjunto de términos: primer orden (FO), interacción de segundo orden (TWI) y segundo orden cuadrático (PQ).

La prueba de hipótesis para cada uno de los coeficientes $\beta_i (i = \overline{0,k})$, incluyendo el intercepto β_0, se plantea de la siguiente forma:

1. Hipótesis nula H_0: El coeficiente β_i es igual a cero.
2. Hipótesis alternativa: El coeficiente β_i es diferente de cero.

Esta última prueba supone que el error se comporta normalmente y el estadístico de prueba *t* se utiliza. *Pr*(>|*t*|) es la significación en dos colas del estadígrafo de la prueba. Si este último es menor que el nivel de significación fijado (α), entonces se rechaza Ho.

En particular, para el caso de la simulación que utiliza el modelo TN93, cuya la salida se muestra en la tabla *A5.1* del Anexo 5, el análisis de varianza (*Tabla 3.1*) indica que el término PQ del segundo orden contribuye significativamente al modelo y la pérdida de ajuste (*Lack of fit*) no es significativa. Obsérvese, además, el análisis de significación de los coeficientes (*Tabla 3.2*), fijándose en la última columna de esta tabla se puede ver que el valor *Pr*(>|*t*|) de los coeficientes del intercepto y del término x_2^2 son menores que el valor prefijado $\alpha = 0.05$. Esto es equivalente a decir que se rechaza la hipótesis nula para los coeficientes correspondientes al intercepto y al término x_2^2 del modelo. Esto confirma que en este caso de simulación, el modelo de segundo orden ajusta la relación entre la variable respuesta y las variables independientes.

Tabla 3.1. *Análisis de varianza de regresión para el modelo TN93*

	Grados de libertad	**Suma de cuadrados**	**Media de los cuadrados**	**Valor F**	**Pr(> F)**
FO(x1, x2)	2	3828.0	1914.0	4.0261	0.052162
TWI(x1, x2)	1	527.7	527.7	1.1099	0.316887
PQ(x1, x2)	2	4424.9	2212.4	4.6539	0.037268
Residuos	10	4754.0	475.4		
Pérdida del ajuste	6	4678.7	779.8	41.4303	0.074301
Error puro	4	75.3	18.8		

La salida del análisis de ajuste también da resultados del análisis canónico, el cual indica que el punto estacionario de la superficie ajustada está en (0.039, 2.863) y pertenece a la región experimental. Además, ambos valores propios son negativos, lo cual confirma que el punto estacionario es un máximo. De hecho, es posible que se tome aproximadamente el punto $t \approx 0.04$ y $proc \approx 3$ como la condición óptima de la superficie, donde se encuentra el valor óptimo de la variable respuesta *Alignment Score*.

Análogamente, en los casos de simulación que utilizan otros modelos evolutivos, los análisis de ajuste se realizan de la misma forma. Para el modelo K80, el análisis de varianza (*Tabla 3.3*) muestra que el modelo de segundo orden se constituye significativamente por el intercepto y el término TWI, y la pérdida de ajuste no es significativa.

Tabla 3.2. *Análisis de significación de los coeficientes para el modelo TN93*

	Estimación	**Error estándar**	**Valor *t***	**Pr (>|*t*|)**
Intercepto	1152.838	8.326	138.456	<2e-16 (***)
$\mathbf{x_1}$	-1.628	8.444	-0.193	0.8510
$\mathbf{x_2}$	-2.932	8.326	-0.352	0.7321
$\mathbf{x_1 : x_2}$	7.264	6.895	1.054	0.3169
$\mathbf{x_1\text{^}2}$	-6.504	11.347	-0.573	0.5792
$\mathbf{x_2\text{^}2}$	-16.116	5.841	-2.759	0.0202 *

El ajuste del modelo se confirma una vez más mediante el análisis de significación de los coeficientes (*Tabla 3.4*), donde se puede ver que el punto de interacción y el intercepto ambos tienen el valor de *Pr*(>|*t*|) menor que 0.05.

Una vez que se ha determinado el modelo adecuado, es posible obtener la respuesta óptima que se encuentra en la superficie ajustada gracias al análisis canónico de segundo orden, el cual indica que el punto estacionario (0.0406, 3.0733) es un máximo, pues los valores propios son ambos negativos. La condición óptima que produce el máximo valor de la variable respuesta puede ser determinada aproximadamente con $t \approx 0.04$ y $proc \approx 3$.

Tabla 3.3. *Análisis de varianza de regresión para el modelo K80*

	Grados de libertad	**Suma de cuadrados**	**Media de los cuadrados**	**Valor F**	**Pr(> F)**
FO(x1, x2)	2	15134	7567	1.5672	0.25582
TWI(x1, x2)	1	26739	26739	5.5382	0.04041
PQ(x1, x2)	2	4898	26739	0.5072	0.61687
Residuos	10	48281	4828		
Pérdida del ajuste	6	48196	8033	379.1631	0.08441
Error puro	4	85	21		

En el caso del modelo JC69, se obtiene también que el modelo de segundo orden ajuste la superficie con la significación de la interacción TWI y el intercepto. La pérdida de ajuste en este caso tampoco es significativa (*Tabla 3.5*).

Tabla 3.4. *Análisis de significación de los coeficientes para el modelo K80*

	Estimación	**Error estándar**	**Valor *t***	**Pr (>|*t*|)**
Intercepto	1127.823	26.535	42.503	1.25e-12 ***
$\mathbf{x_1}$	16.987	26.911	0.631	0.5420
$\mathbf{x_2}$	4.080	26.535	0.154	0.8808
$\mathbf{x_1 : x_2}$	-51.710	21.973	-2.353	0.0404 *
$\mathbf{x_1\text{^}2}$	36.313	36.161	1.004	0.3390
$\mathbf{x_2\text{^}2}$	-3.258	18.613	-0.175	0.8645

La *Tabla 3.6* muestra que los valores de $Pr(>|t|)$ para aquellos términos del modelo están por debajo del punto 0.05, confirmando que el modelo de segundo orden ajusta la superficie. El punto estacionario obtenido está ubicado en (0.0215, 3.9764) y es un máximo, debido a que los valores propios son negativos. La condición óptima de la superficie en este caso se aproxima al punto $t \approx 0.02$ y $proc \approx 4$, donde se logra el valor máximo de la variable respuesta.

Tabla 3.5. *Análisis de varianza de regresión para el modelo JC69*

	Grados de libertad	**Suma de cuadrados**	**Media de los cuadrados**	**Valor F**	**Pr(> F)**
FO(x1, x2)	2	1486.80	743.40	7.0120	0.01250
TWI(x1, x2)	1	920.83	920.83	8.6856	0.01461
PQ(x1, x2)	2	23.18	11.59	0.1093	0.89751
Residuos	10	1060.18	106.02		
Pérdida del ajuste	6	548.16	91.36	0.7137	0.66151
Error puro	4	512.02	128.01		

Para la simulación bajo el modelo evolutivo SG09, se obtuvo que la relación entre las variables independientes y la respuesta se ajuste por el modelo del primer orden, pues sólo el intercepto y el término x_2 tienen coeficientes significativos (*Tabla 3.8*) y, además, el análisis de varianza mostrada en la *Tabla 3.7* indica que el modelo tiene la pérdida de ajuste no significativo.

Tabla 3.6. *Análisis de significación de los coeficientes para el modelo JC69*

	Estimación	**Error estándar**	**Valor *t***	**Pr (>|*t*|)**
Intercepto	1138.7351	3.9320	289.604	<2e-16 ***
$\mathbf{x_1}$	-3.5595	3.9878	-0.893	0.3930
$\mathbf{x_2}$	7.0107	3.9320	1.783	0.1049
$\mathbf{x_1 : x_2}$	9.5960	3.2560	2.947	0.0146 *
$\mathbf{x_1 \text{^} 2}$	2.3426	5.3585	0.437	0.6713
$\mathbf{x_2 \text{^} 2}$	-0.7433	2.7582	-0.269	0.7930

El punto estacionario obtenido a través del análisis canónico se encuentra en (0.038, 2.875), el cual se comporta como un máximo, siendo negativos los valores propios. Luego, las variables independientes se aproximan tomando los valores $t \approx 0.04$ y $proc \approx 3$ para formar la mejor condición de la superficie que maximice el valor de la respuesta.

Tabla 3.7. *Análisis de varianza de regresión para el modelo SG09*

	Grados de libertad	**Suma de cuadrados**	**Media de los cuadrados**	**Valor F**	**Pr(> F)**
FO(x1, x2)	2	773.91	386.96	3.5812	0.03716
TWI(x1, x2)	1	2.80	2.80	0.0259	0.87524
PQ(x1, x2)	2	331.02	165.51	1.5318	0.26284
Residuos	10	1080.51	108.05		
Pérdida del ajuste	6	916.90	152.82	3.7361	0.11137
Error puro	4	163.61	40.90		

A diferente de los casos de simulación anteriormente analizados, los análisis de ajuste para el modelo F81 y el HKY85 no lograron a obtener el modelo adecuado para la superficie. Obsérvese las tablas 3.9 y 3.10 que muestran el análisis de varianza y el de significación de los coeficientes del modelo de segundo orden, respectivamente, para el modelo F81, y las tablas 3.11 y 3.12 para el modelo HKY85, es fácil ver que en estos casos la pérdida de ajuste siempre es muy pequeña, o sea, es significativa, y ninguno de los términos del modelo, excepto el intercepto, tienen coeficientes significativos, debido a que todos los valores de $\Pr(>|t|)$ son superiores que el punto fijado $\alpha = 0.05$.

Tabla 3.8. *Análisis de significación de los coeficientes para el modelo SG09*

	Estimación	**Error estándar**	**Valor *t***	**Pr (>|*t*|)**
Intercepto	1140.7425	3.9696	287.372	<2e-16 ***
$\mathbf{x_1}$	-2.0415	4.0259	-0.507	0.6231
$\mathbf{x_2}$	-9.1850	3.9696	-2.314	0.0432 *
$\mathbf{x_1 : x_2}$	0.5295	3.2871	0.161	0.8752
$\mathbf{x_1\text{^}2}$	-7.7537	5.4096	-1.433	0.1823

x_2^2	-1.7150	2.7845	-0.616	0.5517

Estos resultados indican que, al variar los valores de las variables independientes, los valores observados de la variable respuesta se encuentran distribuidos en un entorno muy cercano de una superficie plana. Es decir, el valor esperado de esa variable no es estadísticamente diferente de una constante (el valor del intercepto), y por tanto, no es posible determinar una respuesta óptima, pues la superficie de respuesta es aproximadamente plana.

Los análisis canónicos de ambos casos también devolvieron puntos estacionarios, sin embargo, debido a la falta de ajuste del modelo de segundo orden, el valor de la variable respuesta evaluado en estos puntos no tiene diferencia significativa, con respecto al valor del intercepto.

Tabla 3.9. *Análisis de varianza de regresión para el modelo F81*

	Grados de libertad	**Suma de cuadrados**	**Media de los cuadrados**	**Valor F**	**Pr(> F)**
FO(x1, x2)	2	14000	7000	1.0400	0.388732
TWI(x1, x2)	1	153	153	0.0227	0.883124
PQ(x1, x2)	2	28812	14406	2.1404	0.168362
Residuos	10	67305	6730		
Pérdida del ajuste	6	65881	10980	30.8412	0.002611
Error puro	4	1424	356		

Tabla 3.10. *Análisis de significación de los coeficientes para el modelo F81*

	Estimación	**Error estándar**	**Valor *t***	**Pr (>|*t*|)**
Intercepto	1119.048	31.329	35.719	7.02e-12 ***
x_1	-3.410	31.774	-0.107	0.917
x_2	-4.687	31.329	-0.150	0.884
x_1 : x_2	3.912	25.943	0.151	0.883
x_1^2	70.852	42.695	1.660	0.128
x_2^2	-35.383	21.977	-1.610	0.138

De todo lo anterior, se puede concluir que para las simulaciones que utilizan el modelo F81 y HKY85, la condición óptima de la superficie no se encuentra en la región experimental o, se dice de otra forma, el espacio de búsqueda queda muy alejado del entorno donde se ubican los valores de las variables independientes que optimizan la variable respuesta.

Hasta aquí, hemos realizado los diseños experimentales y los análisis de ajuste del modelo de segundo orden para las simulaciones bajo diferentes modelos evolutivos de Makov. Se desea ahora que, a partir de los resultados obtenidos, se encuentra el modelo apropiado que constituye la mejor simulación en la predicción de las variantes mutacionales. Esto se logra mediante el método de evaluar la exactitud de los modelos en la reconstrucción de la matriz de tasas de sustitución.

Tabla 3.11. *Análisis de varianza de regresión para el modelo HKY85*

	Grados de libertad	**Suma de cuadrados**	**Media de los cuadrados**	**Valor F**	**Pr(> F)**
FO(x1, x2)	2	21637	10819	1.4021	0.2906
TWI(x1, x2)	1	783	783	0.1015	0.7565
PQ(x1, x2)	2	37296	18648	2.4168	0.1392
Residuos	10	77160	7716		
Pérdida del ajuste	6	76960	12827	255.8165	4.04e-05
Error puro	4	201	50		

Tabla 3.12. *Análisis de significación de los coeficientes para el modelo HKY85*

	Estimación	**Error estándar**	**Valor *t***	**Pr (>|*t*|)**
Intercepto	1121.152	33.545	33.422	1.36e-11 ***
$\mathbf{x_1}$	7.070	34.021	0.208	0.8396
$\mathbf{x_2}$	-5.366	33.545	-0.160	0.8761
$\mathbf{x_1 : x_2}$	8.851	27.778	0.319	0.7565
$\mathbf{x_1\text{^}2}$	74.840	45.714	1.637	0.1326
$\mathbf{x_2\text{^}2}$	-43.036	23.531	-1.829	0.0973 .

En la *Tabla 10.1* presentada en el anexo 10, se muestra un resumen de los valores óptimos obtenidos de la puntuación de alineamiento (*Score*) y los errores relativos (*e*) en la estimación de la matriz de tasas correspondientes a los modelos de Markov. Obsérvese el caso del modelo JC69, se puede ver que el valor del error relativo es 0 para todas las pruebas experimentales de la simulación. Esto se explica basándose en el hecho de que este modelo, conocido como el más sencillo, asume que todos los nucleótidos tienen la misma tasa de sustitución y la matriz de tasas se construye de forma tal que todos los elementos, menos los encontrados en la diagonal, tienen el mismo valor (Ver la sección 1.4.3.1 del Capítulo 1). Por esta razón, no tiene sentido la aplicación del método de evaluación de la exactitud en la reconstrucción de la matriz de tasas para este modelo y, por lo tanto, el mismo se descarta de la comparación de los valores del error relativo entre los modelos. Al comparar estos últimos valores, menos el del modelo JC69, se detectó fácilmente que el TN93 es el mejor modelo, debido a que el error relativo en este caso es menor que los demás. De hecho, podemos concluir finalmente que, para la simulación sin restricciones, el modelo evolutivo TN93 resulta más eficiente en la obtención de una buena predicción de las variantes mutacionales. La simulación logró, además, la mejor condición evolutiva de los parámetros ($t \approx 0.04$, $proc \approx 3$), donde se maximiza el valor de la puntuación de alineamiento de las secuencias de ADN producidas y de las secuencias encontradas en la base de datos Influenza Virus Resources.

3.4.2 Diseños experimentales para la simulación de poblaciones virales bajo la presión selectiva del sistema inmune

En este caso de simulación, los diseños experimentales se llevan a cabo de la misma forma al caso expuesto anteriormente (Ver Anexo 7). En la última columna de los mismos se presentan los valores medio de la puntuación de alineamiento de las variantes mutacionales generadas en las pruebas experimentales, que tienen mayor frecuencia relativa en la población. El objetivo es determinar el modelo más adecuado que ajuste la relación funcional de las variables incluidas en la simulación. Para evaluar la calidad del modelo de ajuste, se complementa cada diseño experimental con cinco réplicas del experimento en el punto central. La aplicación del método *rsm* en *R,* usando el modelo de segundo orden SO, da resultados para diferentes modelos evolutivos que se muestran en las tablas del Anexo 8. Como el caso de simulación sin

restricciones, también son interesados solamente los análisis de varianza y análisis de significación de los coeficientes del modelo.

Por ser parecidos los análisis de ajuste para diferentes modelos de Markov y con el objetivo de evitar la repetición de la misma discusión, en lo siguiente se realizará brevemente el análisis de los resultados para obtener finalmente el modelo más apropiado para la simulación.

El primer modelo que aparece a continuación es el TN93, el cual tiene los resultados de análisis de ajuste mostrados en las tablas 3.13 y 3.14. Observando los valores de *Pr(>F)* y *Pr(>t)* se puede ver que en este caso la pérdida de ajuste para el modelo de segundo orden no es significativa y los términos que constituye el modelo incluyen el intercepto y el cuadrático puro x_2^2. De esta forma, se concluye que este modelo ajusta la superficie de respuesta y un análisis canónico es necesario para encontrar el punto óptimo de la misma. Al realizar este análisis se obtuvo que el punto estacionario (0.0388, 3.0982) es un máximo, pues los valores propios son ambos negativos. Este punto se aproxima tomando $t \approx 0.04$ y $proc \approx 3$ que forma la condición óptima para la superficie donde se obtiene el valor máximo de la variable respuesta.

Tabla 3.13. *Análisis de varianza de regresión para el modelo TN93*

	Grados de libertad	**Suma de cuadrados**	**Media de los cuadrados**	**Valor F**	**Pr(> F)**
FO(x1, x2)	2	86578	43289	24.6824	0.0001356
TWI(x1, x2)	1	4840	4840	2.7595	0.1276596
PQ(x1, x2)	2	48453	24226	13.8133	0.0013259
Residuos	10	17538	1754		
Pérdida del ajuste	6	17464	2911	156.5057	0.107063
Error puro	4	74	19		

Tabla 3.14. *Análisis de significación de los coeficientes para el modelo TN93*

	Estimación	**Error estándar**	**Valor *t***	**Pr (>|*t*|)**
Intercepto	1155.656	15.993	72.261	6.28e-15 ***
$\mathbf{x_1}$	3.834	16.220	0.236	0.817909
$\mathbf{x_2}$	10.226	15.993	0.639	0.536926
$\mathbf{x_1 : x_2}$	-22.000	13.243	-1.661	0.127660
$\mathbf{x_1 \text{^} 2}$	14.479	21.794	0.664	0.521481
$\mathbf{x_2 \text{^} 2}$	-58.506	11.218	-5.215	0.000393 ***

La misma situación ocurre para simulación bajo el modelo K80, pues se obtuvo el ajuste de la relación entre las variables independientes y la respuesta de la simulación mediante el modelo de segundo orden (Ver tablas 3.15 y 3.16), el cual se constituye significativamente por el término PQ de x_2^2 y el FO del intercepto. El punto máximo encontrado de la superficie tiene las coordenadas (0.0395, 3.0673) que pueden ser aproximadas para las variables *t* y *proc* de forma tal que $t \approx 0.04$ y $proc \approx 3$, en el cual el valor de la puntuación de alineamiento se maximiza.

Tabla 3.15. *Análisis de varianza de regresión para el modelo K80*

	Grados de libertad	**Suma de cuadrados**	**Media de los cuadrados**	**Valor F**	**Pr(> F)**
FO(x1, x2)	2	11960.0	5980.0	29.0112	6.867e-05
TWI(x1, x2)	1	839.0	839.0	4.0701	0.071283
PQ(x1, x2)	2	7262.7	3631.4	17.6170	0.000528
Residuos	10	2061.3	206.1		
Pérdida del ajuste	6	1904.4	317.4	8.0943	0.131319
Error puro	4	156.9	39.2		

Tabla 3.16. *Análisis de significación de los coeficientes para el modelo K80*

	Estimación	**Error estándar**	**Valor *t***	**Pr (>|*t*|)**
Intercepto	1152.014	5.483	210.117	< 2e-16 ***
$\mathbf{x_1}$	1.024	5.560	0.184	0.857572
$\mathbf{x_2}$	2.833	5.483	0.517	0.616549
$\mathbf{x_1 : x_2}$	-9.160	4.540	-2.017	0.071283 .
$\mathbf{x_1}$**^2**	8.050	7.472	1.077	0.306582
$\mathbf{x_2}$**^2**	-22.774	3.846	-5.922	0.000147 ***

La simulación con el modelo evolutivo JC69 una vez más resulta la superficie ajustada, mediante el modelo de segundo orden que se comporta por la interacción TWI y el intercepto. En este caso, la pérdida de ajuste tampoco es significativa (Tablas 3.17 y 3.18). El punto estacionario se encuentra en (0.0591, 2.9403) y se toma el punto aproximado ($t \approx 0.06$, $proc \approx 3$) como el máximo donde se logra el valor óptimo de la superficie.

Tabla 3.17. *Análisis de varianza de regresión para el modelo JC69*

	Grados de libertad	**Suma de cuadrados**	**Media de los cuadrados**	**Valor F**	**Pr(> F)**
FO(x1, x2)	2	7620.8	3810.4	3.5762	0.067355
TWI(x1, x2)	1	7046.9	7046.9	6.6138	0.027808
PQ(x1, x2)	2	1069.4	534.7	0.5018	0.619883
Residuos	10	10654.8	1065.5		
Pérdida del ajuste	6	10262.4	1710.4	17.4368	0.077143
Error puro	4	392.4	98.1		

Tabla 3.18. *Análisis de significación de los coeficientes para el modelo JC69*

	Estimación	**Error estándar**	**Valor *t***	**Pr (>|*t*|)**
Intercepto	1146.291	12.465	91.959	5.66e-16 ***
$\mathbf{x_1}$	2.827	12.642	0.224	0.8276
$\mathbf{x_2}$	11.487	12.465	0.922	0.3785
$\mathbf{x_1 : x_2}$	-26.546	10.322	-2.572	0.0278 *
$\mathbf{x_1}$**^2**	-8.792	16.987	-0.518	0.6160
$\mathbf{x_2}$**^2**	-6.137	8.744	-0.702	0.4988

Los resultados obtenidos en el análisis de ajuste para la simulación bajo el modelo SG09 (Tablas 3.19 y 3.20) indican que el modelo formado por la interacción TWI, el cuadrático puro PQ correspondiente al término x_2^2 y el intercepto, es adecuado que ajuste la relación funcional entre las variables incluidas en la simulación. El punto máximo de la superficie está ubicado en (0.0407, 3.1903), del cual se toman $t \approx 0.04$ y $proc \approx 3$.

Tabla 3.19. *Análisis de varianza de regresión para el modelo SG09*

	Grados de libertad	**Suma de cuadrados**	**Media de los cuadrados**	**Valor F**	**Pr(> F)**
FO(x1, x2)	2	19276.9	9638.4	12.9681	0.0016685
TWI(x1, x2)	1	8357.0	8357.0	11.2440	0.0073256
PQ(x1, x2)	2	7698.1	3849.0	5.1787	0.0286014
Residuos	10	7432.4	743.2		
Pérdida del ajuste	6	7351.0	1225.2	60.2023	0.709423
Error puro	4	81.4	20.4		

Tabla 3.20. *Análisis de significación de los coeficientes para el modelo SG09*

	Estimación	**Error estándar**	**Valor *t***	**Pr (>|*t*|)**
Intercepto	1151.147	10.411	110.570	< 2e-16 ***
$\mathbf{x_1}$	-13.745	10.559	-1.302	0.22220
$\mathbf{x_2}$	3.469	10.411	0.333	0.74583
$\mathbf{x_1 : x_2}$	28.908	8.621	3.353	0.00733 **
$\mathbf{x_1 \text{^} 2}$	7.636	14.188	0.538	0.60222
$\mathbf{x_2 \text{^} 2}$	-23.421	7.303	-3.207	0.00938 **

Las tablas 3.21 y 3.22 muestran que, para la simulación con el modelo markoviano F81, el modelo de segundo orden no ajusta, igual que el caso de simulación sin restricciones anterior. Para la superficie de este caso tampoco se puede encontrar la condición óptima, debido a que el valor esperado de la variable respuesta no es estadísticamente diferente de una constante (el valor del intercepto) y, por lo tanto, la superficie de respuesta es aproximadamente plana.

Tabla 3.21. *Análisis de varianza de regresión para el modelo F81*

	Grados de libertad	**Suma de cuadrados**	**Media de los cuadrados**	**Valor F**	**Pr(> F)**
FO(x1, x2)	2	2485.2	1242.6	1.9391	0.194231
TWI(x1, x2)	1	2140.2	2140.2	3.3399	0.097568
PQ(x1, x2)	2	1951.3	975.7	1.5225	0.264706
Residuos	10	6408.1	640.8		
Pérdida del ajuste	6	6316.8	1052.8	46.1414	0.001194
Error puro	4	91.3	22.8		

Tabla 3.22. *Análisis de significación de los coeficientes para el modelo F81*

	Estimación	**Error estándar**	**Valor *t***	**Pr (>|*t*|)**
Intercepto	1146.5794	9.6670	118.607	<2e-16 ***
$\mathbf{x_1}$	2.3265	9.8041	0.237	0.8172
$\mathbf{x_2}$	0.1197	9.6670	0.012	0.9904
$\mathbf{x_1 : x_2}$	-14.6295	8.0050	-1.828	0.0976 .
$\mathbf{x_1}$**^2**	-10.3943	13.1739	-0.789	0.4484
$\mathbf{x_2}$**^2**	-8.8894	6.7811	-1.311	0.2192

Sin embargo, al diferente de la simulación bajo el modelo evolutivo HKY85 previamente discutida, en este caso se obtuvo el modelo de segundo orden que ajuste la superficie de respuesta y el mismo se comporta como un polinomio compuesto por el término del primer orden (FO) correspondiente al término x_2, la interacción (TWI), el cuadrático puro (PQ) de x_2^2, y el intercepto (Ver Tablas 3.23 y 3.24). La pérdida de ajuste es grande y no es significativa. El máximo de la puntuación de alineamiento se obtiene en el punto (0.0403, 3.0662), del cual se aproximan las variables independientes de la simulación de modo que $t \approx 0.04$, $proc \approx 3$.

Tabla 3.23. *Análisis de varianza de regresión para el modelo HKY85*

	Grados de libertad	**Suma de cuadrados**	**Media de los cuadrados**	**Valor F**	**Pr(> F)**
FO(x1, x2)	2	215.37	107.68	2.5132	0.13053
TWI(x1, x2)	1	220.52	220.52	5.1468	0.04668
PQ(x1, x2)	2	335.50	167.75	3.9151	0.05549
Residuos	10	428.47	42.85		
Pérdida del ajuste	6	109.34	18.22	0.2284	0.94625
Error puro	4	319.13	79.78		

Tabla 3.24. *Análisis de significación de los coeficientes para el modelo evolutivo HKY85*

	Estimación	**Error estándar**	**Valor *t***	**Pr (>|*t*|)**
Intercepto	1136.530	2.500	454.665	<2e-16 ***
$\mathbf{x_1}$	-1.610	2.535	-0.635	0.5398
$\mathbf{x_2}$	7.369	2.500	2.948	0.0146 *
$\mathbf{x_1 : x_2}$	4.696	2.070	2.269	0.0467 *
$\mathbf{x_1 \text{^} 2}$	-4.293	3.407	-1.260	0.2362
$\mathbf{x_2 \text{^} 2}$	4.793	1.753	2.734	0.0211 *

Al igual que el caso de simulación en ausencia de restricciones previamente realizada, se quiere encontrar también el modelo evolutivo adecuado que mejore la predicción de las variantes mutacionales. De nuevo esto se hace mediante la evaluación de la corrección de los modelos evolutivos en la reconstrucción de la matriz de tasas de sustitución, de forma tal que mientras tenga menor el valor del error relativo, será mejor el modelo. Obsérvese el Anexo 10, donde se muestran los valores óptimos de la puntuación de alineamiento logrados para los modelos evolutivos y los errores relativos de la estimación de la matriz de tasas. Fijándose la parte correspondiente al caso de la simulación en presencia de la presión selectiva del sistema inmune, teniendo en cuenta que, en este caso de simulación, se descarta también el modelo JC69 pues, como se ha explicado en la sección anterior, el mismo no tiene sentido para la aplicación del método de evaluación de la exactitud en la estimación de la matriz de tasas. De hecho, se determinó fácilmente, una vez más, que el modelo TN93 resulta mejor para la simulación, debido al valor del error relativo es menor que el de los demás modelos.

En base de que la simulación use el modelo TN93 para mejorar la predicción de las variantes mutacionales, se descubrió que la condición óptima de los parámetros, que maximice el valor de la puntuación de alineamiento de las secuencias de ADN generadas, es $t \approx 0.04$ y $proc \approx 3$. Este resultado confirma que, tanto para la simulación sin restricciones como la simulación bajo la presión selectiva, el modelo TN93 constituye siempre la mejor opción que forma parte esencial para el algoritmo de simulación, capaz de predecir eficientemente las variantes mutacionales del virus. La obtención de este resultado también es de espera pues, en comparación con los demás modelos de cuatro bases, el TN93 es el más general que asume

menos restricciones sobre la matriz de tasas de transiciones. Lo cual trae, como consecuencia, la flexibilidad en el uso de este modelo para el proceso de sustitución de nucleótidos, que contribuye a mejorar la efectividad de la simulación.

3.5 Conclusiones parciales del capítulo

Se expusieron los resultados obtenidos de las pruebas experimentales ejecutadas en Linux para ambos casos de simulación, en ausencia o en presencia de la presión selectiva del sistema inmune. Mostrando, a través de la comparación de los mismos, las diferencias en cuanto al tamaño de las poblaciones generadas y, además, el efecto de los parámetros de la simulación en la distribución de frecuencias de los alelos, y en el valor predictivo de la misma.

Se realizaron también los diseños experimentales, mediante el método de superficie de respuesta (RSM) disponible en *R*, sobre los resultados de las pruebas experimentales realizadas, con el propósito de encontrar la mejor combinación de los parámetros incluidos en la simulación, que optimice la puntuación de alineamiento de las secuencias mutantes generadas.

Se obtuvieron, para algunos modelos evolutivos, las superficies de respuesta donde se pudo determinar la condición óptima de las variables independientes, que maximiza el valor de la variable respuesta (*Alignment Score*). Sin embargo, para otros modelos, no se logró a encontrar en la región experimental el punto óptimo, pues las superficies en estos casos resultaron aproximadamente planas. A pesar de esto, se obtuvieron siempre altos porcientos de identidades entre las secuencias generadas y las encontradas en la base de datos Influenza Virus Resources, lo cual es un indicativo positivo de la consistencia del procedimiento desarrollado.

La evaluación de la exactitud de los modelos markovianos en la reconstrucción de la matriz de tasas, mediante el error relativo, da lugar un método eficiente para descubrir el modelo más adecuado para la simulación. De esta forma, se obtuvo, en ambos casos de simulación, que el modelo TN93 es el mejor, entre otros, capaz de predecir bien las variantes mutacionales en el proceso evolutivo de las poblaciones virales.

CONCLUSIONES

El análisis de los resultados obtenidos de las pruebas experimentales realizadas, mediante la aplicación del algoritmo de simulación sobre las poblaciones virales, conduce a las siguientes conclusiones:

- La simulación de Monte Carlo, utilizando los modelos evolutivos Markovianos, sugiere el empleo factible de este procedimiento en la predicción *in silico* de variantes mutacionales de escapes que evaden la acción del sistema inmune, inducida por la vacunación.
- El uso de la metodología de diseños experimentales Superficie Respuesta resultó eficiente en detectar la mejor condición de los parámetros evolutivos que optimiza el valor predictivo de la simulación.
- Mediante la evaluación de la exactitud en la reconstruccion de la matriz de tasas de transiciones, se detectó, en cualqier caso de simulacion, que el modelo TN93 siempre es el más adecuado, entre los demás, que mejor predice las variantes mutacionales.

RECOMENDACIONES

1. Los trabajos de aplicación de la simulación de mutaciones de la Hemaglutinina (HA) del virus de la Influenza A/H1N1 debieran ser extendidos a otras proteínas del virus, en particular la Neuronamidasa (NA) porque ella está presente también el drift antigénico y es importante identificar, las mutaciones posibles en dicho sitio, que pueden reducir o inhibir la unión de anticuerpos neutralizantes, en particular inhibidores de la propia NA, como son el oseltamivir y el zanamavir (¿por qué conformarnos, o al menos conformarnos solo con el Tamiflú?)

2. Trabajos similares de aplicación son deseables, y pueden ser extendidos a otros virus, en particular otras cepas de la influenza como el H5N1, el H7N1, la peste porcina clásica (PPC), todas de interés, por ejemplo del CENSA (Centro Nacional de Sanidad Agropecuaria) o el VIH (de interés local, nacional o inclusive internacional).

3. Se debe tratar, en particular de encadenar investigaciones que articulen el pronóstico de mutaciones de proteínas del VIH con los nuevos modelos evolutivos, el análisis de la resistencia antiviral y el estudio de respuesta del sistema inmune, para cumplimentar estudios a "ciclo completo" para beneficio de Cuba, Cambridge, Europa, Vietnam y el mundo…

REFERENCIAS BIBLIOGRÁFICAS

GENTLE, J. E., HARDLE, W., MORI, Y. & WANG, Y. (2004) *Handbook of Computational Statistics*, Springer Heidelberg.

NEI, M. (1975) *Molecular population genetics and evolution*, North-Holland publishing company - Amsterdam, Oxford.

YANG, Z. (2006) *Computational Molecular Evolution* Oxford University Press Inc., New York

ANDRIEU, C., N. D. F., ARNAUD D., MICHAEL I. J., (2003) An Introduction to MCMC for Machine Learning.

E. GULTEPE, M. L. K. (2005). "Monte Carlo simulation and statistical analysis of genetic information coding." Science Direct.

JUAN CARLOS SALAZAR, A. B. Z. (2008). "Análisis y diseño de experimentos aplicados a estudios de simulación."

KLEIJNEN, J. P. C. (2001). "Experimental designs for sensitivity analysis of simulation models".

MONTGOMERY and DOUGLAS (1997). "Design and analysis of experiments". New York, John Wiley & Sons.

ROBERT, C. and G. CASELLA (2004). "Monte Carlo Statistical Methods". New York, Springer.

SANCHEZ, R. and R. GRAU (2008). "Extended genetic code vector space over the Galois field of five DNA bases alphabet. An hypothesis about the primeval genetic code." In Press.

GIBERGANS, J. (2003) "Regresión lineal simple".

MORELISI (2003) "Modelo de regresión lineal simple".

GRIFFITHS, A. J. F., MILLER, J. H., SUZUKI, D. T., LEWONTIN, R. C., and GELBART, W. M. (2000). *An Introduction to Genetic Analysis*. W.H. Freeman and Company. Chapter 10 (Molecular Biology of Gene Function): Genetic code: Stop codons

SANCHEZ, R. & GRAU, R. (2009) "An algebraic hypothesis about the primeval genetic code architecture". *Mathematical Biosciences,* 221.

Baldi, P. and S. Brunak (2001). Bioinformatics. The machine learning approach. London, England, The MIT Press.

Beerli, P. (2006). "STATISTICAL METHODS IN MOLECULAR EVOLUTION."

Brent, R. P. (1973). Algorithms for Minimizations without Derivatives. New Jersey, Prentice-Hall.

Brian S., Torsten H. (2009). A Handbook of Statistical Analyses using R. London, CRC Press.

Dan Graur, W.-H. L. (2000). Fundamentals of Molecular Evolution, Sinauer Associates.

Desper, R. a. O. G. (2002). "Fast and Accurate Phylogeny Reconstruction Algorithms Based on the Minimum-Evolution Principle." Journal of Computational Biology **9**: 687-705.

Douglas, C. (2005). "Response surface Methods from Design and Analysis of Experiments ".

Felsenstein, J. (1981). "Evolutionary Trees from DNA Sequences: A Maximum Likelihood Approach." Journal of Molecular Evolution **17**: 368-376.

Fitch, W. (1971). "Toward defining the course of evolution: minimum change for a specific tree topology." Systematic Zoology **20**: 406-416.

Keyuan X., G. C. V., Peter C. Doerschuk (2005). Stochastic Modeling of Biological Sequence Evolution. Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology.

Kimmel, M. (2006). "Statistical Methods in Molecular Evolution." Journal of the American Statistical Association **101**.

Kumar S., A. F. (2008). "Molecular Phylogeny Reconstruction." Encyclopedia of Life Sciences (ELS).

Lenth, R. V. (2009). "Response-surface Analysis."

Lenth, R. V. (2009). "Response-Surface Methods in R, Using rsm." Journal of Statistical Software.

Marco S., A.-M. V. (1998). The Phylogenetic Handbook. A practical Approach to DNA and Protein Phylogeny.

Mederos O.; Grau, R. H. R. G. L. A. (1986). "“Ecuaciones Diferenciales Ordinarias”." Pueblo y Educación.

Nielsen, R. (2005). Statistical Methods in Molecolar Evolution.

Press, W. H., S. A. Teukolsky, et al. (1992). Numerical Recipes in C: The art of scientific computing. Cambridge, Cambridge University Press.

Robert C. E., G. A., Serafim B. and Arend S. (2009). "Evolver: a whole-genome sequence evolution simulator."

Saitou, N. a. M. N. (1987). "The Neighbor-joining Method: A New Method for Reconstructing Phylogenetic Trees." Molecular Biology and Evolution **4**: 406-425.

Sánchez R. , G. R. (2009). "Nuevos Modelos algebraicos y markovianos del proceso evolutivo. Aplicaciones a la predicción de la Influenza." Conferencia Magistral en COMPUMAT , La Habana, Cuba.

Shao-Min Yan, G. W. (2009). "Prediction of mutation position, mutated amino acid and timing in hemagglutinins from North America H1 influenza A virus." J. Biomedical Science and Engineering, **2**.

TAMURA K, D. J., M, N. & S, K. (2007). "MEGA4: Molecular Evolutionary Genetics Analysis (MEGA) software version 4.0." Mol Biol Evol **24**: 1596-1599.

W.J. Ewens, G. R. G. (2001). Statistical Methods in Bioinformatics, an Introduction. New York, Springer.

Yang, Z. (2006). Computational Molecular Evolution Oxford University Press Inc., New York

Yang Z., T. W. (1995). "Mixed Model Analysis of DNA sequence Evolution." Biometrics **51**: 552-561.

ANEXOS

Anexo 1. Base de datos usada para las pruebas experimentales de la simulación

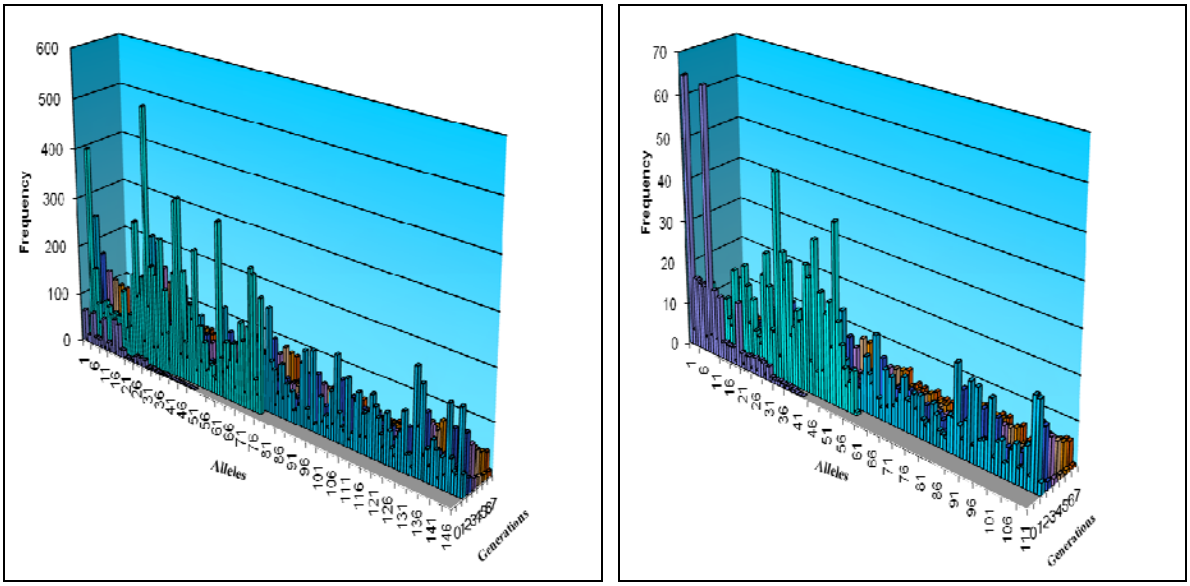
Tabla A1. *Descripción de la base de datos*

Número	Nombre de la especie	Código de acceso
1	A/Texas/05/2009(H1N1)	gb|GQ457487
2	A/Vladivostok/01/2009(H1N1)	gb|GU211219
3	A/Ontario/05/2009(H1N1)	gb|CY076795
4	A/Saint-Petersburg/RII35/2009(H1N1)	gb|CY075477
5	A/England/04/2009(H1N1)	gb|CY057051
6	A/Nevada/05/2009(H1N1)	gb|GQ894821
7	A/Singapore/05/2009(H1N1)	gb|CY049372
8	A/Nebraska/02/2009(H1N1)	gb|GQ377082
9	A/New York/04/2009(H1N1)	gb|GQ160568
10	A/Rio de Janeiro/05/2009(H1N1)	gb|CY054280
11	A/Toronto/03/2009(H1N1)	gb|FJ974026
12	A/Mexico City/02/2009(H1N1)	gb|CY064740
13	A/Argentina/06/2009(H1N1)	gb|CY053907
14	A/Canadá-AB/RV1644/2009(H1N1)	gb|GQ465679
15	A/Brawley/05/2009(H1N1)	gb|CY073394
16	A/San Diego/03/2009(H1N1)	gb|CY073366
17	A/Arizona/06/2009(H1N1)	gb|GQ232037
18	A/Thailand/02/2009(H1N1)	gb|CY075815
19	A/Nizhny Novgorod/01/2009(H1N1)	gb|CY075483
20	A/South Dakota/04/2009(H1N1)	gb|GQ231990
21	A/Tver/03/2009(H1N1)	gb|HM101145
22	A/Kuwait/03/2009(H1N1)	gb|CY062430
23	A/Idaho/02/2009(H1N1))	gb|GQ221826
24	A/Qingdao/03/2009(H1N1)	gb|CY050265

25	A/Brownsville/06/2009(H1N1)	gb|CY053254
26	A/Craven/01/2009(H1N1)	gb|CY049820
27	A/Colorado/04/2009(H1N1)	gb|GQ221820
28	A/Afghanistan/01/2009(H1N1)	gb|CY062402
29	A/Hawaii/06/2009(H1N1)	gb|GQ338358
30	A/California/01/2009(H1N1)	gb|GQ323483
31	A/Managua/01/2009(H1N1)	gb|CY073585
32	A/Sao Paulo/05/2009(H1N1)	gb|GQ915018
33	A/Mexico/04/2009(H1N1)	gb|GQ132145
34	A/Wisconsin/06/2009(H1N1)	gb|CY055415
35	A/Sachsen-Anhalt/01/2009(H1N1)	gb|CY045482
36	A/Norway/05/2009(H1N1)	gb|CY051984
37	A/Paris/02/2009(H1N1)	gb|GQ329100
38	A/Houston/02/2009(H1N1)	gb|CY053135
39	A/Kuwait/05/2009(H1N1)	gb|CY062388
40	A/Moscow/02/2009(H1N1)	gb|HM189396
41	A/Ivanovo/01/2009(H1N1)	gb|HM189387
42	A/Georgia/01/2009(H1N1)	gb|GQ200237
43	A/Finland/04/2009(H1N1)	gb|HQ228040
44	A/California/02/2009(H1N1)	gb|CY054827
45	A/Afghanistan/05/2009(H1N1)	gb|CY062378
46	A/Shizuoka-C/06/2009(H1N1)	gb|GU014804
47	A/New York/03/2009(H1N1)	gb|CY051543
48	A/Texas/04/2009(H1N1)	gb|CY052807
49	A/Canadá-MB/05/2009(H1N1)	gb|GQ402201
50	A/Brownsville/03/2009(H1N1)	gb|CY053206
51	A/Wisconsin/03/2009(H1N1)	gb|CY052114
52	A/Mexico City/06/2009(H1N1)	gb|CY062522
53	A/Moscow oblast/02/2009(H1N1)	gb|HM189411

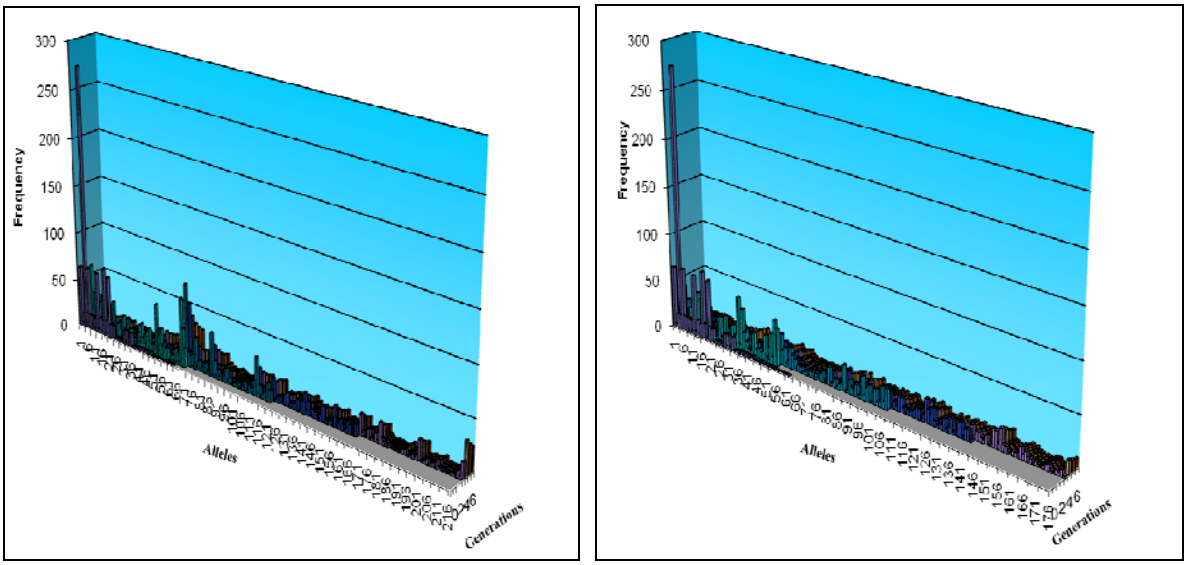
54	A/Ontario/01/2009(H1N1)	gb|CY076763
55	A/Wisconsin/06/2009(H1N1)	gb|CY051295
56	A/Brawley/02/2009(H1N1)	gb|CY043094
57	A/Afghanistan/04/2009(H1N1)	gb|CY062391
58	A/Ankara/04/2009(H1N1)	gb|GU369647
59	A/San Diego/06/2009(H1N1)	gb|CY073371
60	A/Houston/04/2009(H1N1)	gb|CY053079
61	A/Canada-QC/04/2009(H1N1)	gb|GQ402199
62	A/Texas/06/2009(H1N1)	gb|CY052367
63	A/Kuwait/03/2009(H1N1)	gb|CY062423
64	A/Ulyanovsk/05/2009(H1N1)	gb|HM189405
65	A/Brawley/01/2009(H1N1)	gb|CY043086

Anexo 2. Representación gráfica de los resultados obtenidos en ambos casos de simulación. A: en ausencia de presión selectiva, B: en presencia de presión selectiva del sistema inmune.



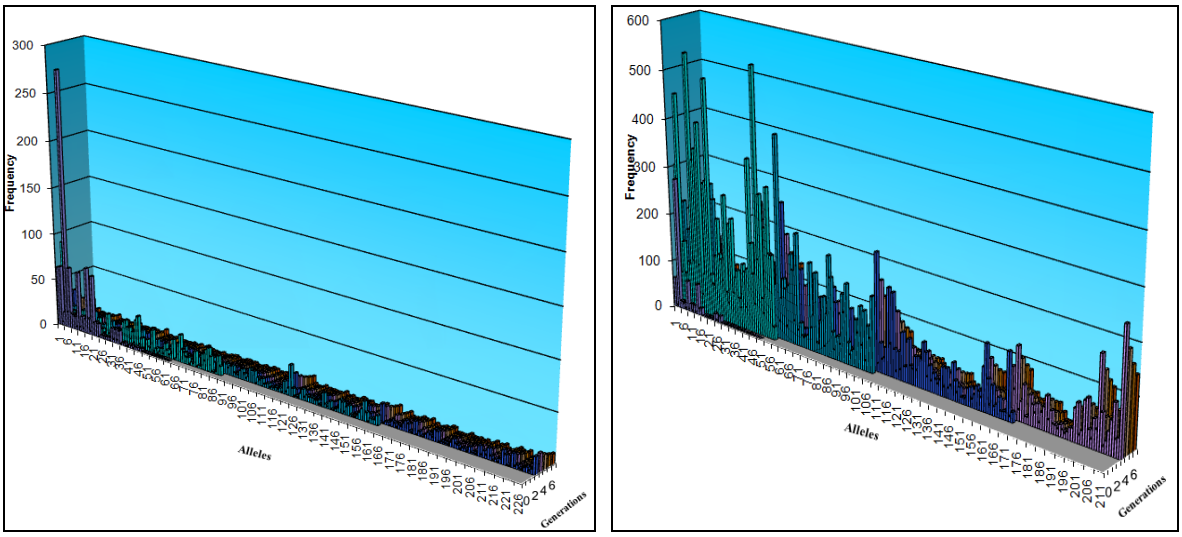
A B

Figura A2.1. *Gráficos de las poblaciones virales simuladas con los parámetros t = 0.02, proc = 2 y m = TN93.*



A B

Figura A2.2. *Gráficos de las poblaciones virales simuladas con los parámetros t = 0.04, proc = 3 y m = TN93.*



A B

***Figura A2.3**. Gráficos de las poblaciones virales simuladas con los parámetros t = 0.06, proc = 2 y m = TN93.*

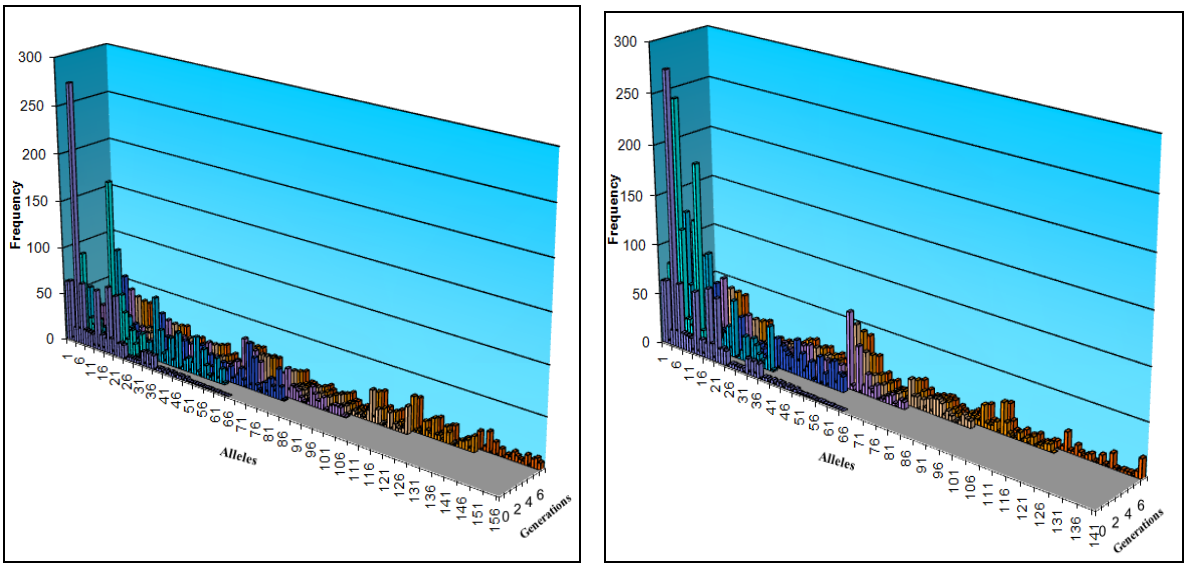
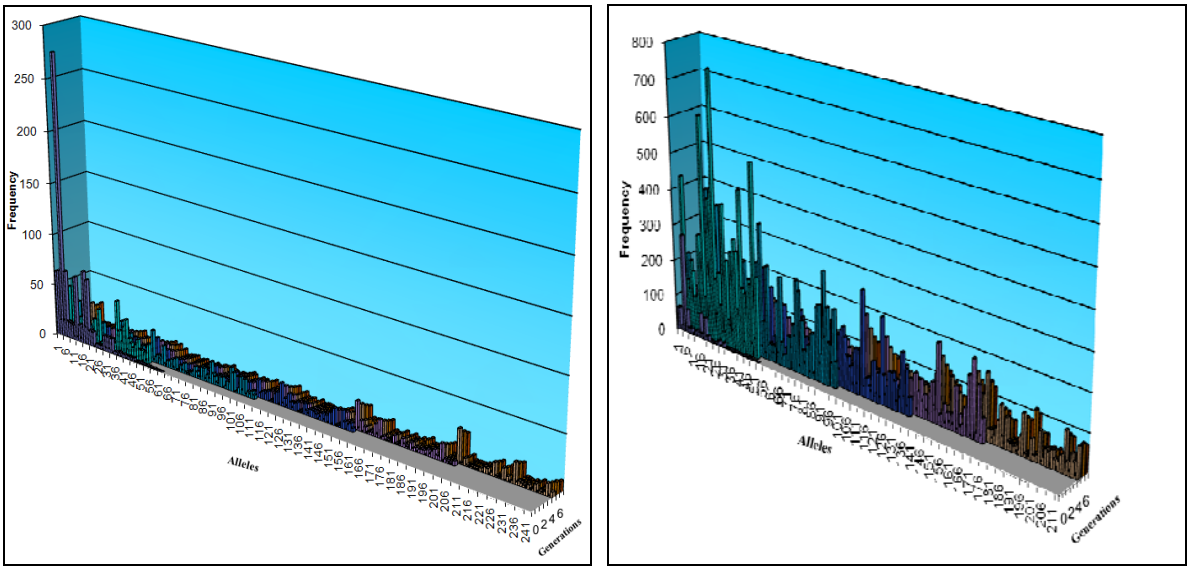


Figura A2.4. *Gráficos de las poblaciones virales simuladas con los parámetros t = 0.02, proc = 3 y m = K80.*



A B

Figura A2.5. *Gráficos de las poblaciones virales simuladas con los parámetros t = 0.04, proc = 3 y m = K80.*

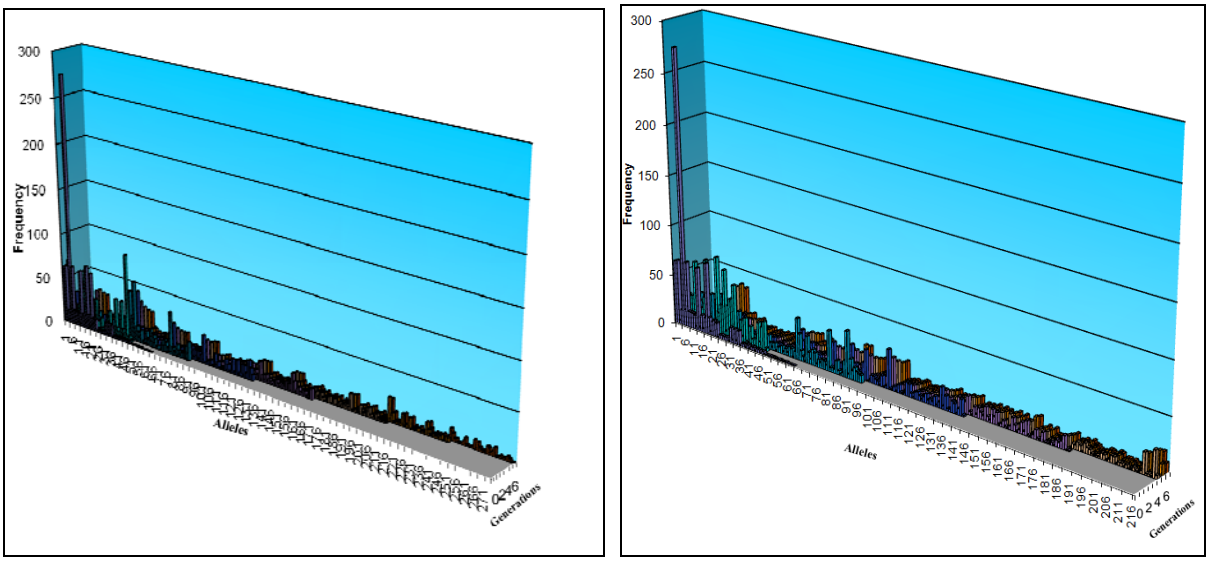


***Figura A2.6**. Gráficos de las poblaciones virales simuladas con los parámetros t = 0.06, proc = 4 y m = K80.*

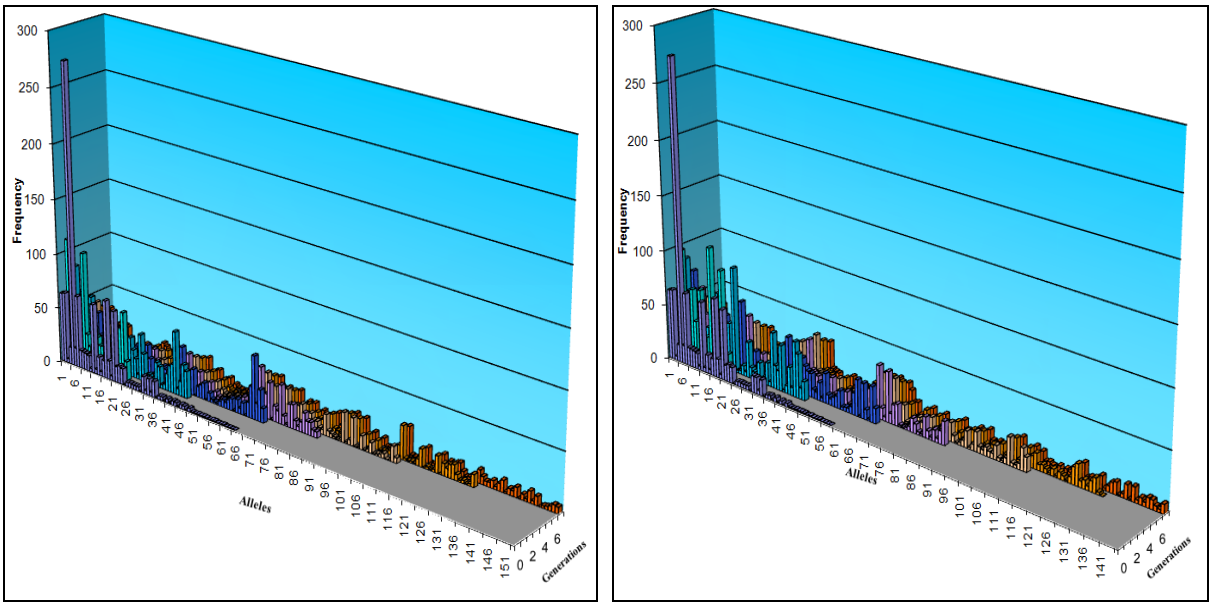
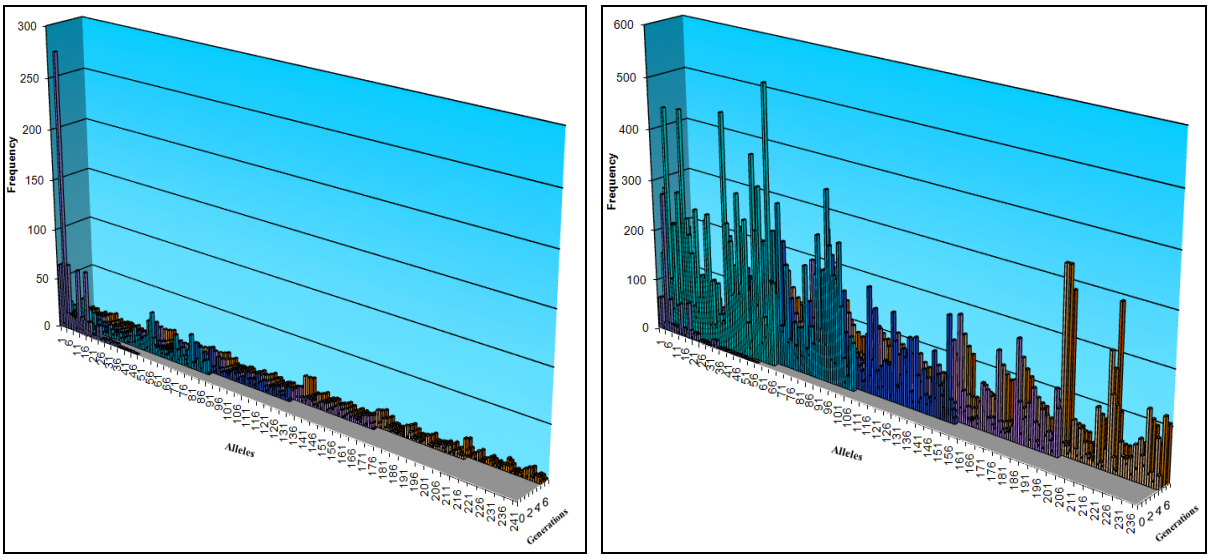
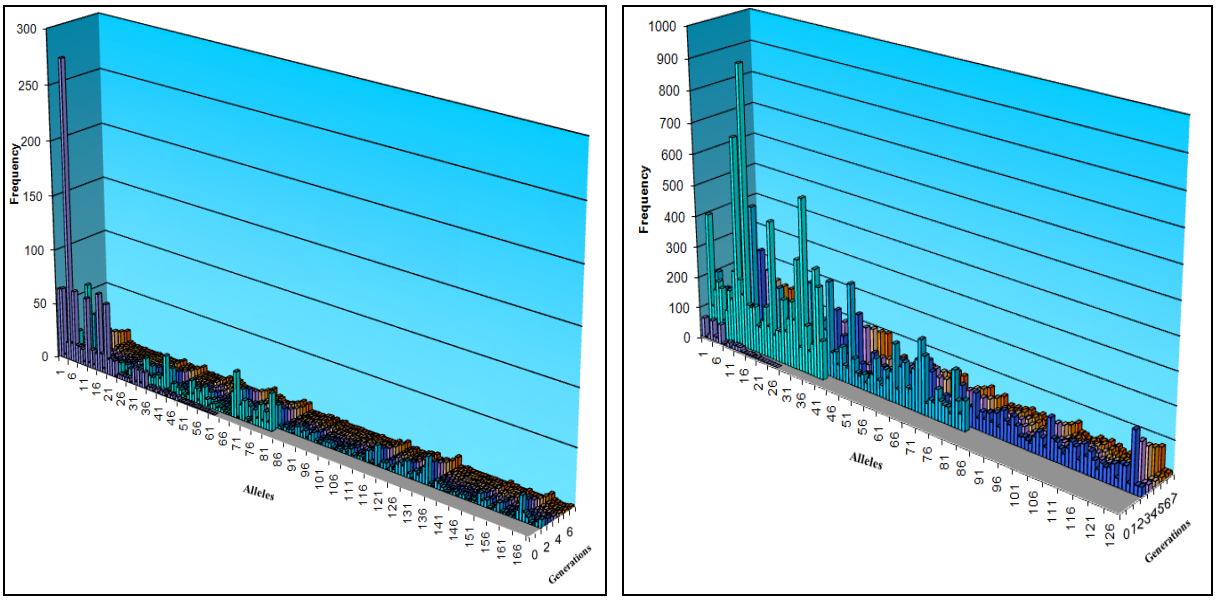


***Figura A2.7**. Gráficos de las poblaciones virales simuladas con los parámetros t = 0.02, proc = 2 y m = JC69.*



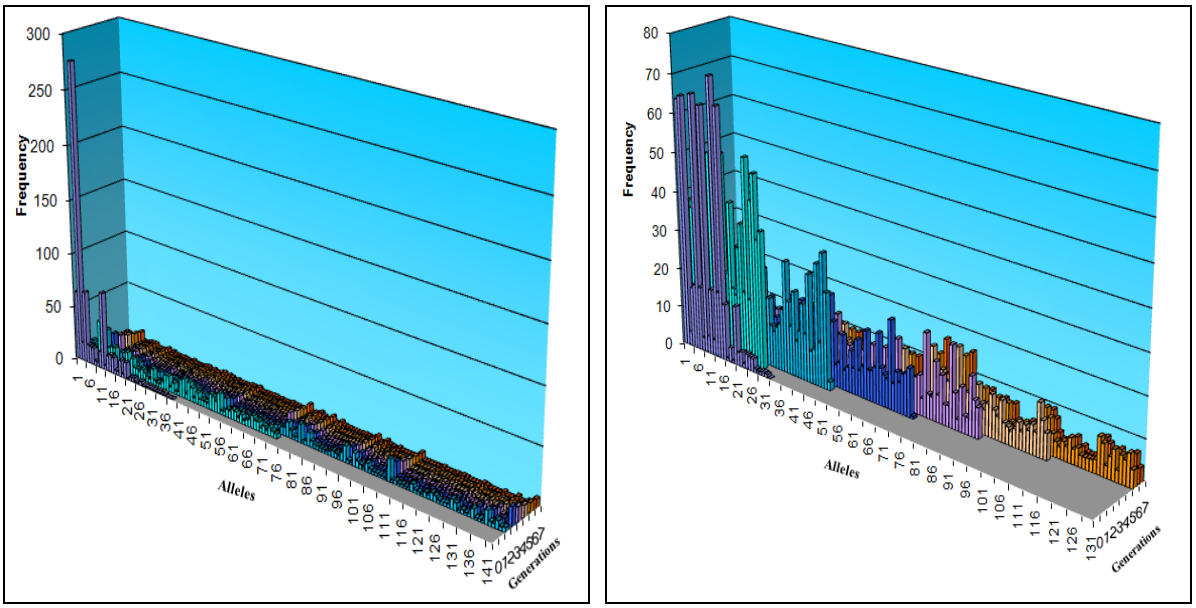
A B

***Figura A2.8**. Gráficos de las poblaciones virales simuladas con los parámetros t = 0.04, proc = 3 y m = JC69.*



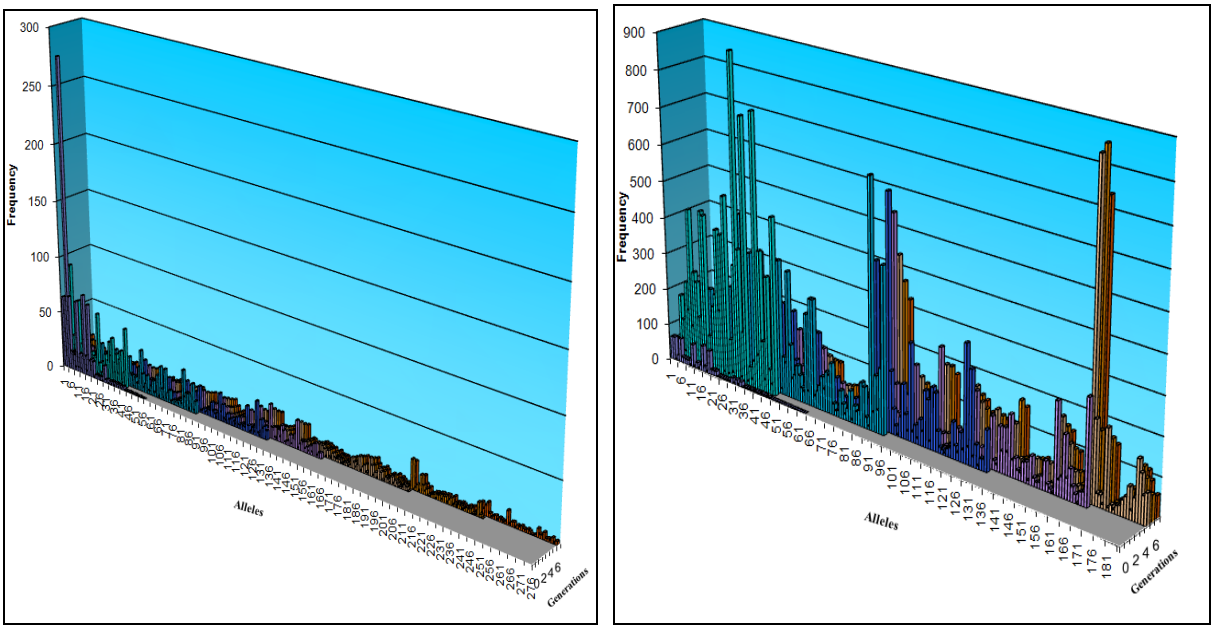
A B

***Figura A2.9**. Gráficos de las poblaciones virales simuladas con los parámetros t = 0.06, proc = 4 y m = JC69.*



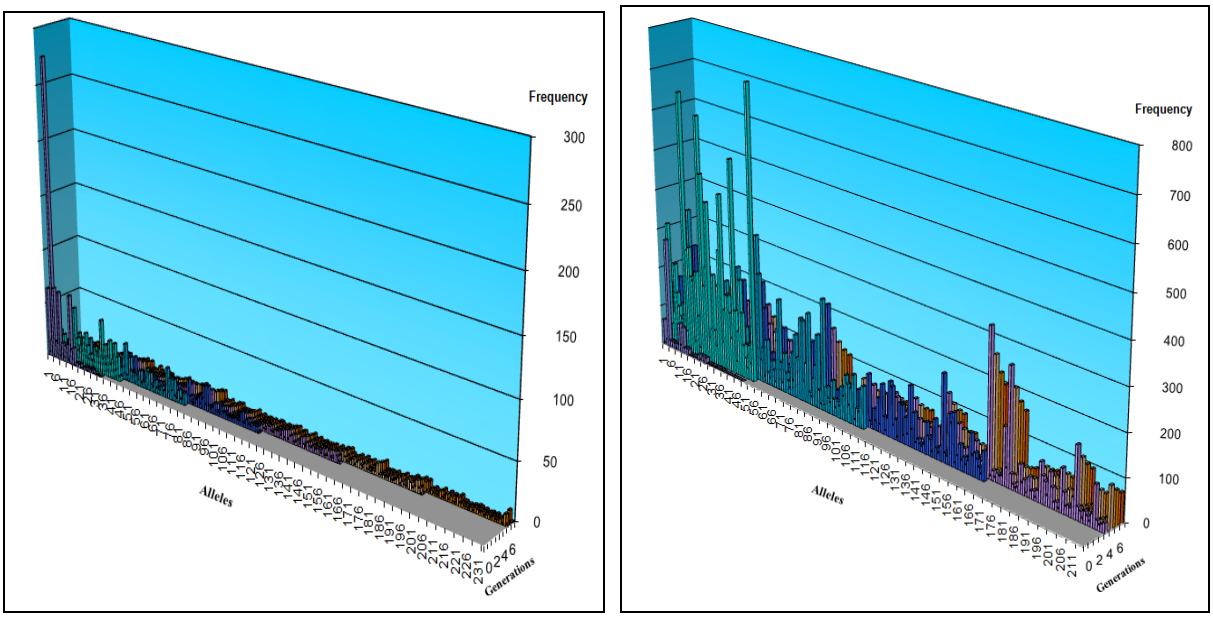
A B

Figura A2.10. *Gráficos de las poblaciones virales simuladas con los parámetros t = 0.02, proc = 3 y m = SG09.*



A B

Figura A2.11. *Gráficos de las poblaciones virales simuladas con los parámetros t = 0.04, proc = 3 y m = SG09.*



A B

Figura A2.12. *Gráficos de las poblaciones virales simuladas con los parámetros t = 0.06, proc = 3 y m = SG09.*

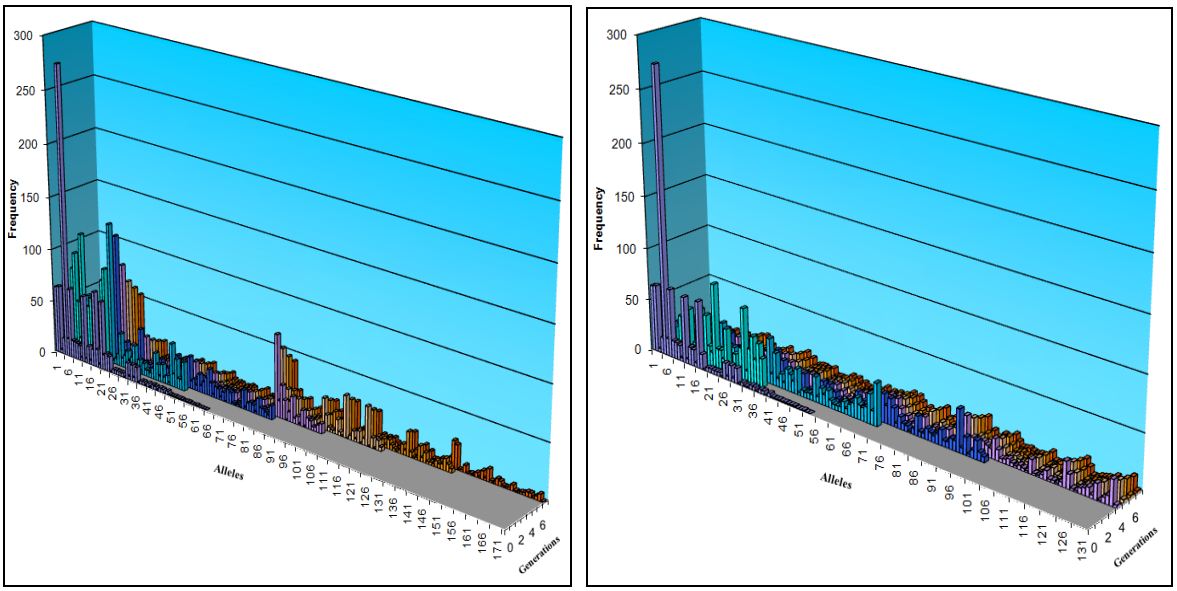


***Figura A2.13**. Gráficos de las poblaciones virales simuladas con los parámetros t = 0.02, proc = 2 y m = F81.*

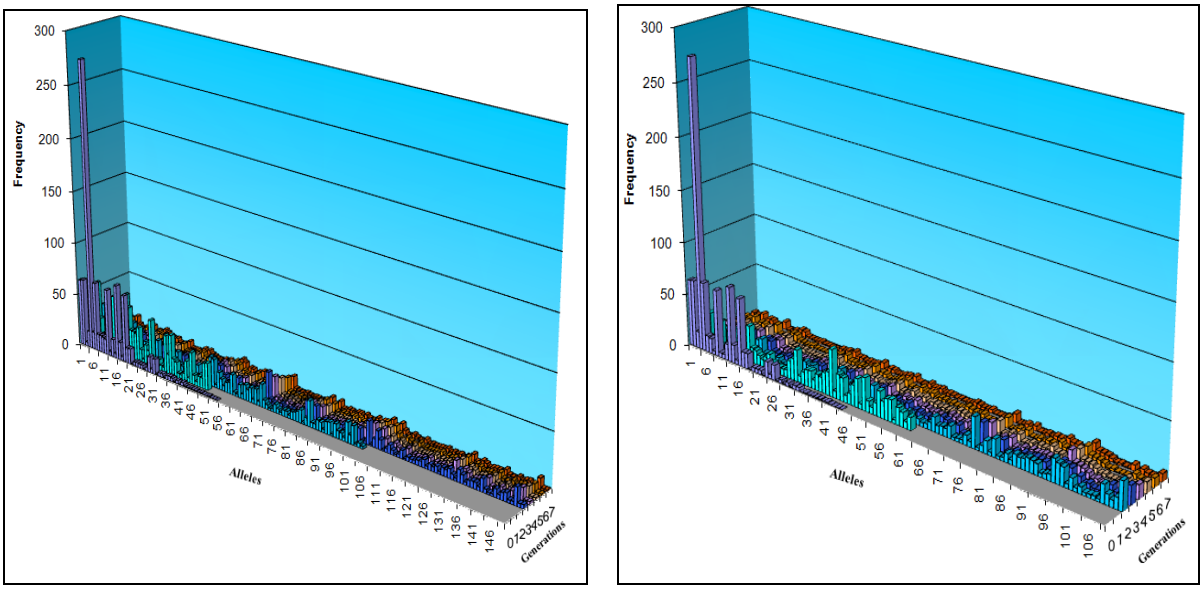
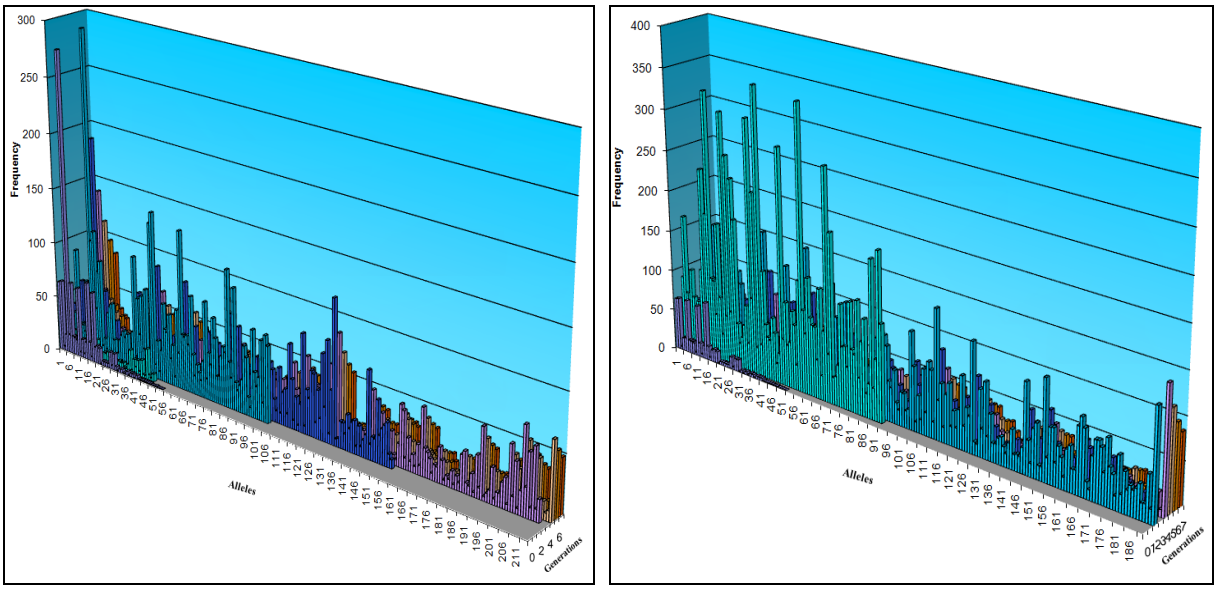
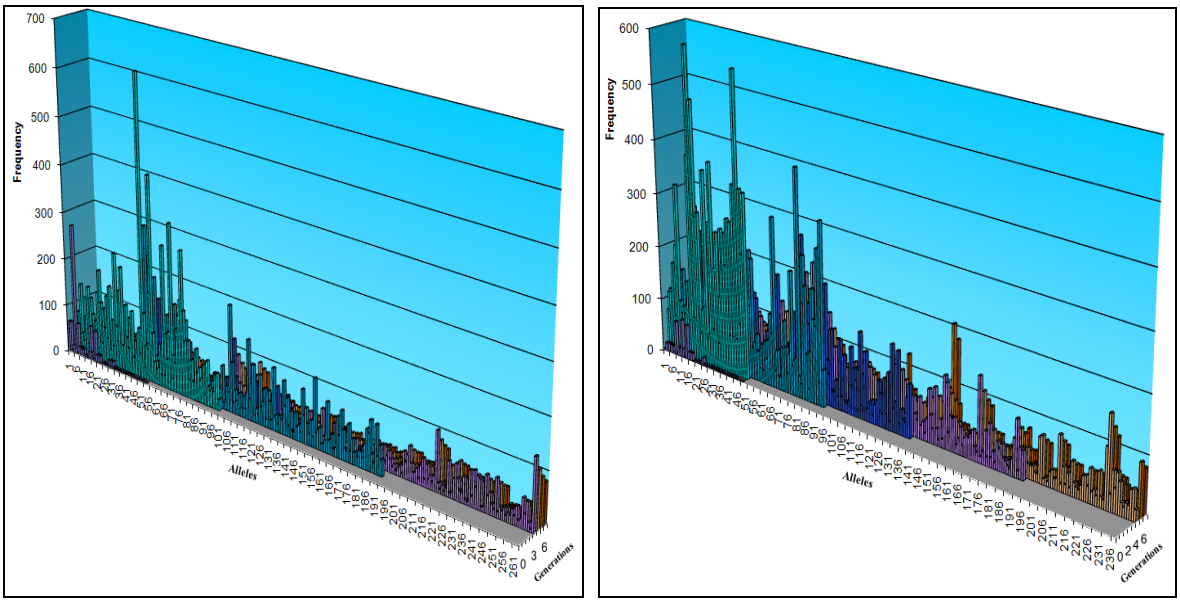


Figura A2.14. *Gráficos de las poblaciones virales simuladas con los parámetros t = 0.04, proc = 3 y m = F81.*



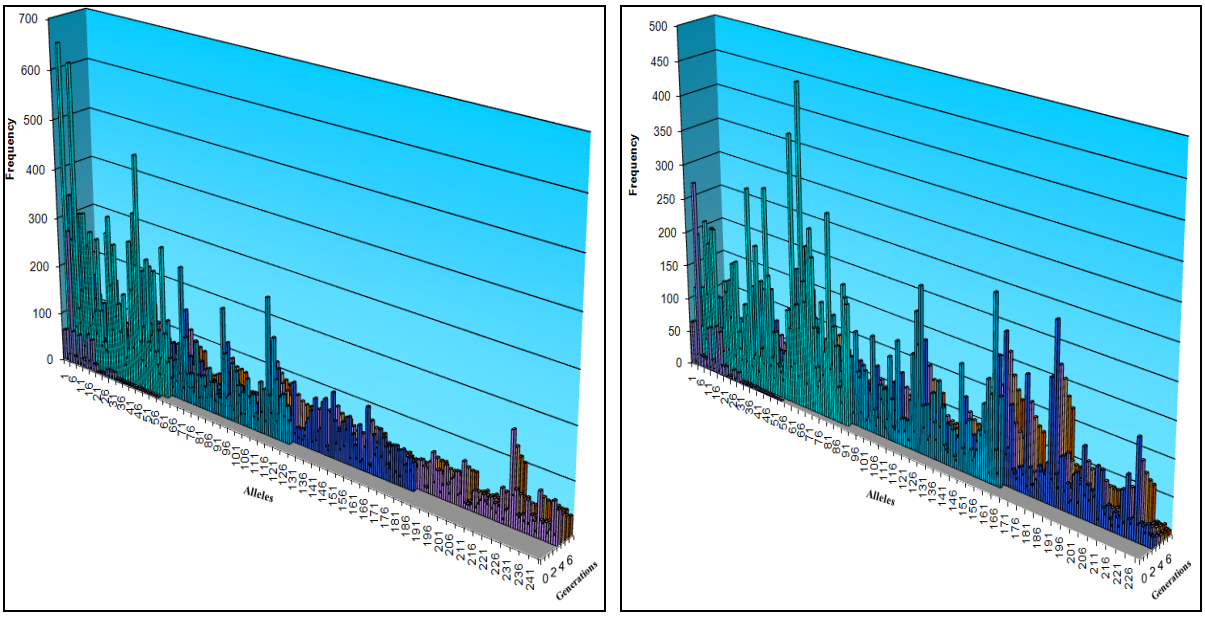
A B

Figura A2.15. *Gráficos de las poblaciones virales simuladas con los parámetros t = 0.06, proc = 4 y m = F81.*



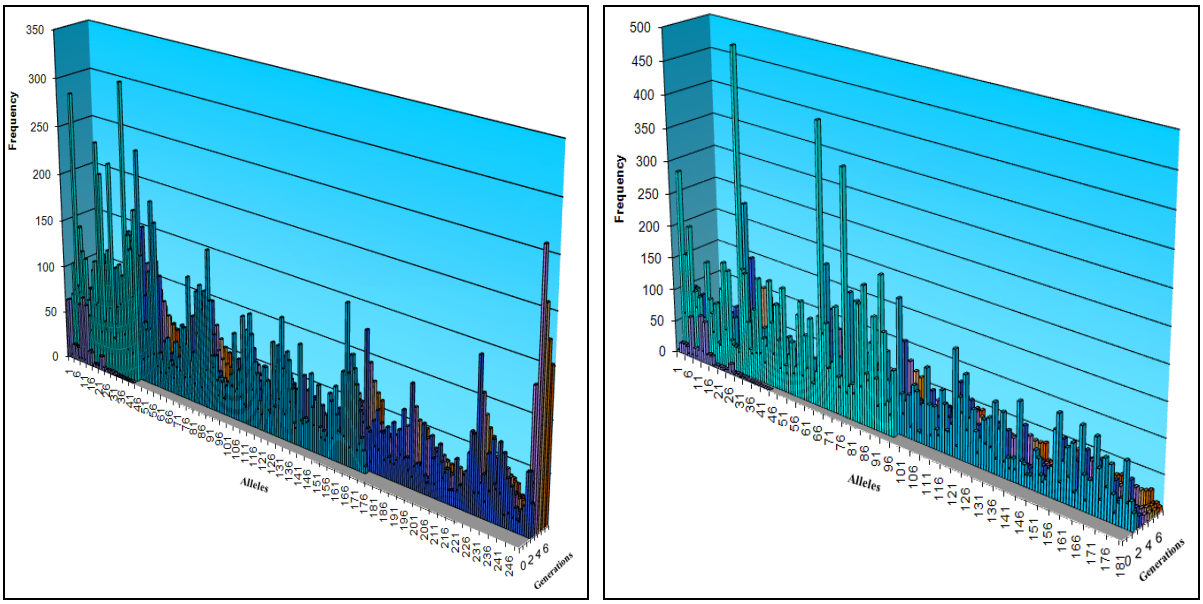
A B

Figura A2.16. *Gráficos de las poblaciones virales simuladas con los parámetros t = 0.02, proc = 4 y m = HKY85.*



A B

Figura A2.17*. Gráficos de las poblaciones virales simuladas con los parámetros t = 0.04, proc = 3 y m = HKY85.*



A B

Figura A2.18. *Gráficos de las poblaciones virales simuladas con los parámetros t = 0.06, proc = 2 y m = HKY85.*

Anexo 3. Número de alelos producidos en las pruebas experimentales

Tabla A3. *Comparación del número de alelos generados entre dos casos de simulación*

Parámetros de simulación			Número de alelos generados	
m	*t*	*proc*	Simulación sin restricciones	Simulación bajo presión selectiva
TN93	0.02	1	149	115
TN93	0.04	1	261	195
TN93	0.06	1	241	211
TN93	0.02	2	145	134
TN93	0.04	2	328	272
TN93	0.06	2	509	478
TN93	0.02	3	156	138
TN93	0.04	3	380	328
TN93	0.06	3	592	563
TN93	0.02	4	421	305
TN93	0.04	4	755	476
TN93	0.06	4	844	735
K80	0.02	1	121	102
K80	0.04	1	220	205
K80	0.06	1	370	321
K80	0.02	2	142	122
K80	0.04	2	323	316
K80	0.06	2	539	456
K80	0.02	3	157	141
K80	0.04	3	360	357

K80	0.06	3	570	563
K80	0.02	4	207	189
K80	0.04	4	247	217
K80	0.06	4	275	250
JC69	0.02	1	117	111
JC69	0.04	1	194	140
JC69	0.06	1	208	180
JC69	0.02	2	153	141
JC69	0.04	2	388	342
JC69	0.06	2	548	522
JC69	0.02	3	144	130
JC69	0.04	3	380	377
JC69	0.06	3	538	491
JC69	0.02	4	113	104
JC69	0.04	4	127	119
JC69	0.06	4	190	132
F81	0.02	1	156	112
F81	0.04	1	188	182
F81	0.06	1	290	268
F81	0.02	2	291	173
F81	0.04	2	420	389
F81	0.06	2	634	593
F81	0.02	3	196	141
F81	0.04	3	357	258
F81	0.06	3	665	523

F81	0.02	4	361	351
F81	0.04	4	594	474
F81	0.06	4	754	624
HKY85	0.02	1	114	105
HKY85	0.04	1	363	282
HKY85	0.06	1	674	496
HKY85	0.02	2	207	205
HKY85	0.04	2	447	355
HKY85	0.06	2	739	608
HKY85	0.02	3	211	198
HKY85	0.04	3	552	415
HKY85	0.06	3	751	624
HKY85	0.02	4	252	155
HKY85	0.04	4	308	280
HKY85	0.06	4	442	397
SG	0.02	1	247	177
SG	0.04	1	325	287
SG	0.06	1	530	375
SG	0.02	2	121	114
SG	0.04	2	325	290
SG	0.06	2	541	460
SG	0.02	3	142	132
SG	0.04	3	326	265
SG	0.06	3	553	452
SG	0.02	4	442	386

SG	0.04	4	632	499
SG	0.06	4	800	621

Anexo 4. Diseños experimentales de la simulación en ausencia de la presión selectiva

Tabla A4.1. *Diseño experimental correspondiente al modelo evolutivo TN93*

Variables naturales		**Variables codificadas**		**Respuesta**
t	*proc*	x_1	x_2	*Alignment Score (% de similitud)*
0.02	1	-1	-2	1079.40 (91%)
0.04	1	0	-2	1140.73 (96%)
0.06	1	1	-2	1041.13 (90%)
0.02	2	-1	-1	1149.87 (97%)
0.04	2	0	-1	1137.67 (96%)
0.06	2	1	-1	1143.73 (96%)
0.02	3	-1	0	1156.07 (97%)
0.06	3	1	0	1147.40 (97%)
0.02	4	-1	1	1130.20 (95%)
0.04	4	0	1	1125.40 (95%)
0.06	4	1	1	1141.20 (96%)
0.04	3	0	0	1138.27 (96%)
0.04	3	0	0	1149.60 (97%)
0.04	3	0	0	1145.67 (97%)
0.04	3	0	0	1147.33 (97%)
0.04	3	0	0	1147.20 (97%)

Tabla A4.2. *Diseño experimental correspondiente al modelo evolutivo K80*

Variables naturales		**Variables codificadas**		**Respuesta**
t	*proc*	x_1	x_2	*Alignment Score (% de similitud)*
0.02	1	-1	-2	1019.80 (87%)
0.04	1	0	-2	996.87 (86%)
0.06	1	1	-2	1083.27 (93%)
0.02	2	-1	-1	1152.93 (97%)
0.04	2	0	-1	1133.33 (95%)
0.06	2	1	-1	1125.53 (95%)
0.02	3	-1	0	1142.00 (96%)
0.06	3	1	0	1137.60 (96%)
0.02	4	-1	1	1142.73 (96%)
0.04	4	0	1	1154.60 (97%)
0.06	4	1	1	1153.80 (97%)
0.04	3	0	0	1135.00 (96%)
0.04	3	0	0	1139.93 (96%)
0.04	3	0	0	1146.47 (97%)
0.04	3	0	0	1143.80 (96%)
0.04	3	0	0	1144.87 (96%)

Tabla A4.3. *Diseño experimental correspondiente al modelo evolutivo JC69*

Variables naturales		**Variables codificadas**		**Respuesta**
t	*proc*	x_1	x_2	*Alignment Score (% de similitud)*
0.02	1	-1	-2	1152.80 (97%)
0.04	1	0	-2	1113.27 (95%)
0.06	1	1	-2	1097.87 (94%)
0.02	2	-1	-1	1147.73 (97%)
0.04	2	0	-1	1133.27 (96%)
0.06	2	1	-1	1134.53 (96%)
0.02	3	-1	0	1135.73 (96%)
0.06	3	1	0	1130.67 (95%)
0.02	4	-1	1	1143.00 (96%)
0.04	4	0	1	1153.33 (97%)
0.06	4	1	1	1149.33 (97%)
0.04	3	0	0	1137.93 (96%)
0.04	3	0	0	1128.07 (95%)
0.04	3	0	0	1126.33 (95%)
0.04	3	0	0	1147.53 (97%)
0.04	3	0	0	1151.67 (97%)

Tabla A4.4. *Diseño experimental correspondiente al modelo evolutivo SG09*

Variables naturales		**Variables codificadas**		**Respuesta**
t	*proc*	x_1	x_2	*Alignment Score (% de similitud)*
0.02	1	-1	-2	1143.13 (96%)
0.04	1	0	-2	1152.27 (97%)
0.06	1	1	-2	1152.07 (97%)
0.02	2	-1	-1	1142.13 (97%)
0.04	2	0	-1	1144.87 (96%)
0.06	2	1	-1	1123.47 (95%)
0.02	3	-1	0	1153.87 (97%)
0.06	3	1	0	1131.40 (95%)
0.02	4	-1	1	1110.13 (95%)
0.04	4	0	1	1133.80 (96%)
0.06	4	1	1	1123.87 (95%)
0.04	3	0	0	1146.07 (96%)
0.04	3	0	0	1140.87 (96%)
0.04	3	0	0	1143.60 (96%)
0.04	3	0	0	1142.87 (96%)
0.04	3	0	0	1129.67 (96%)

Tabla A4.5. *Diseño experimental correspondiente al modelo evolutivo F81*

Variables naturales		**Variables codificadas**		**Respuesta**
t	*proc*	x_1	x_2	*Alignment Score (% de similitud)*
0.02	1	-1	-2	856.80 (77%)
0.04	1	0	-2	798.73 (71%)
0.06	1	1	-2	837.07 (76%)
0.02	2	-1	-1	1151.00 (97%)
0.04	2	0	-1	1146.00 (97%)
0.06	2	1	-1	1139.07 (96%)
0.02	3	-1	0	1148.27 (97%)
0.06	3	1	0	1127.80 (95%)
0.02	4	-1	1	1122.07 (95%)
0.04	4	0	1	1135.07 (96%)
0.06	4	1	1	1131.27 (96%)
0.04	3	0	0	1135.53 (96%)
0.04	3	0	0	1103.27 (93%)
0.04	3	0	0	1137.13 (96%)
0.04	3	0	0	1138.60 (96%)
0.04	3	0	0	1155.13 (97%)

Tabla A4.6. *Diseño experimental correspondiente al modelo evolutivo HKY85*

Variables naturales		**Variables codificadas**		**Respuesta**
t	*proc*	x_1	x_2	*Alignment Score (% de similitud)*
0.02	1	-1	-2	1133.50 (96%)
0.04	1	0	-2	1078.40 (91%)
0.06	1	1	-2	1129.93 (95%)
0.02	2	-1	-1	1146.87 (97%)
0.04	2	0	-1	1139.27 (96%)
0.06	2	1	-1	1135.20 (96%)
0.02	3	-1	0	1158.27 (97%)
0.06	3	1	0	1135.53 (96%)
0.02	4	-1	1	1087.27 (92%)
0.04	4	0	1	1141.33 (96%)
0.06	4	1	1	1146.40 (96%)
0.04	3	0	0	1135.47 (96%)
0.04	3	0	0	1125.93 (95%)
0.04	3	0	0	1145.00 (96%)
0.04	3	0	0	1140.40 (96%)
0.04	3	0	0	1135.93 (96%)

Anexo 5. Resultados obtenidos de los diseños experimentales RSM realizados en R para la simulación en ausencia de la presión selectiva

Tabla A5.1 *Salida del análisis de ajuste para el modelo de segundo orden, en caso del modelo evolutivo TN93*

```
Residuals:
    Min      1Q  Median      3Q     Max
-30.446  -7.474  -2.611   8.151  46.494

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 1152.838      8.326 138.456   <2e-16 ***
x1            -1.628      8.444  -0.193   0.8510
x2            -2.932      8.326  -0.352   0.7321
x1:x2          7.264      6.895   1.054   0.3169
x1^2          -6.504     11.347  -0.573   0.5792
x2^2         -16.116      5.841  -2.759   0.0202 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 21.8 on 10 degrees of freedom
Multiple R-squared: 0.6488,   Adjusted R-squared: 0.4731
F-statistic: 3.694 on 5 and 10 DF,  p-value: 0.03737

Analysis of Variance Table

Response: Score
            Df Sum Sq Mean Sq F value   Pr(>F)
FO(x1, x2)   2 3828.0  1914.0  4.0261 0.052162
TWI(x1, x2)  1  527.7   527.7  1.1099 0.316887
PQ(x1, x2)   2 4424.9  2212.4  4.6539 0.037268
Residuals   10 4754.0   475.4
Lack of fit  6 4678.7   779.8 41.4303 0.074301
Pure error   4   75.3    18.8

Stationary point of response surface:
        x1         x2
-0.2012644 -0.1363159

Stationary point in original units:
         t       proc
0.0397471 2.86368407

Eigen analysis:
$values
[1]  -5.286343 -17.334490

$vectors
           [,1]       [,2]
[1,] -0.9481049 -0.3179576
[2,] -0.3179576  0.9481049
```

Tabla A5.2. *Salida del análisis de ajuste para el modelo de segundo orden, en caso del modelo evolutivo K80*

```
Call:
rsm(formula = Score ~ SO(x1, x2), data = current)

Residuals:
    Min      1Q  Median      3Q     Max
-109.76  -14.74   12.48   19.88  119.92

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 1127.823     26.535  42.503 1.25e-12 ***
x1            16.987     26.911   0.631   0.5420
x2             4.080     26.535   0.154   0.8808
x1:x2        -51.710     21.973  -2.353   0.0404 *
x1^2          36.313     36.161   1.004   0.3390
x2^2          -3.258     18.613  -0.175   0.8645
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 69.48 on 10 degrees of freedom
Multiple R-squared: 0.4921,  Adjusted R-squared: 0.2381
F-statistic: 1.937 on 5 and 10 DF,  p-value: 0.1747

Analysis of Variance Table

Response: Score
             Df Sum Sq Mean Sq  F value    Pr(>F)
FO(x1, x2)    2  15134    7567   1.5672   0.25582
TWI(x1, x2)   1  26739   26739   5.5382   0.04041
PQ(x1, x2)    2   4898    2449   0.5072   0.61687
Residuals    10  48281    4828
Lack of fit   6  48196    8033 379.1631  0.084405
Pure error    4     85      21

Stationary point of response surface:
        x1         x2
0.03186764 0.37327194

Stationary point in original units:
         t       proc
0.04063735 3.07327194

Eigen analysis:
$values
[1]  -49.08392 -16.02966

$vectors
           [,1]       [,2]
[1,] -0.8965826 -0.4428765
[2,]  0.4428765 -0.8965826
```

Tabla A5.3. *Salida del análisis de ajuste para el modelo de segundo orden, en caso del modelo evolutivo JC69*

```
Call:
rsm(formula = Score ~ SO(x1, x2), data = current)

Residuals:
     Min       1Q   Median       3Q      Max
-12.4051  -7.2539   0.2228   6.5558  14.3617

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 1138.7351     3.9320 289.604   <2e-16 ***
x1            -3.5595     3.9878  -0.893   0.3930
x2             7.0107     3.9320   1.783   0.1049
x1:x2          9.5960     3.2560   2.947   0.0146 *
x1^2           2.3426     5.3585   0.437   0.6713
x2^2          -0.7433     2.7582  -0.269   0.7930
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 10.3 on 10 degrees of freedom
Multiple R-squared: 0.6963,   Adjusted R-squared: 0.5445
F-statistic: 4.586 on 5 and 10 DF,  p-value: 0.01958

Analysis of Variance Table

Response: Score
            Df  Sum Sq Mean Sq F value  Pr(>F)
FO(x1, x2)   2 1486.80  743.40  7.0120 0.01250
TWI(x1, x2)  1  920.83  920.83  8.6856 0.01461
PQ(x1, x2)   2   23.18   11.59  0.1093 0.89751
Residuals   10 1060.18  106.02
Lack of fit  6  548.16   91.36  0.7137 0.66151
Pure error   4  512.02  128.01

Stationary point of response surface:
        x1         x2
-0.6257877  0.6764787

Stationary point in original units:
         t       proc
0.02148425 3.97647866

Eigen analysis:
$values
[1]  -5.839670 -4.240318

$vectors
           [,1]       [,2]
[1,] -0.8081286  0.5890060
[2,] -0.5890060 -0.8081286
```

Tabla A5.4. *Salida del análisis de ajuste para el modelo de segundo orden, en caso del modelo evolutivo SG09*

```
Call:
rsm(formula = Score ~ SO(x1, x2), data = current)

Residuals:
     Min       1Q   Median       3Q      Max
-14.4177  -3.6242   0.2901   3.4593  18.8397

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 1140.7425     3.9696 287.372   <2e-16 ***
x1            -2.0415     4.0259  -0.507   0.6231
x2            -9.1850     3.9696  -2.314   0.0432 *
x1:x2          0.5295     3.2871   0.161   0.8752
x1^2          -7.7537     5.4096  -1.433   0.1823
x2^2          -1.7150     2.7845  -0.616   0.5517
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 10.39 on 10 degrees of freedom
Multiple R-squared: 0.5062,   Adjusted R-squared: 0.2593
F-statistic:  2.05 on 5 and 10 DF,  p-value: 0.1563

Analysis of Variance Table

Response: Score
            Df  Sum Sq Mean Sq F value  Pr(>F)
FO(x1, x2)   2  773.91  386.96  3.5812 0.03716
TWI(x1, x2)  1    2.80    2.80  0.0259 0.87524
PQ(x1, x2)   2  331.02  165.51  1.5318 0.26284
Residuals   10 1080.51  108.05
Lack of fit  6  916.90  152.82  3.7361 0.11137
Pure error   4  163.61   40.90

Stationary point of response surface:
        x1         x2
-0.0742624 -0.1246266

Stationary point in original units:
         t       proc
0.03851475  2.8753734

Eigen analysis:
$values
[1] -1.703415 -7.765335

$vectors
            [,1]        [,2]
[1,] -0.04371608 -0.99904400
[2,] -0.99904400  0.04371608
```

Tabla A5.5. *Salida del análisis de ajuste para el modelo de segundo orden, en caso del modelo evolutivo F81*

```
Call:
rsm(formula = Score ~ SO(x1, x2), data = current)

Residuals:
     Min       1Q   Median       3Q      Max
-188.159  -21.112    1.835   41.084   90.564

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 1119.048     31.329  35.719 7.02e-12 ***
x1            -3.410     31.774  -0.107    0.917
x2            -4.687     31.329  -0.150    0.884
x1:x2          3.912     25.943   0.151    0.883
x1^2          70.852     42.695   1.660    0.128
x2^2         -35.383     21.977  -1.610    0.138
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 82.04 on 10 degrees of freedom
Multiple R-squared: 0.3896,   Adjusted R-squared: 0.08445
F-statistic: 1.277 on 5 and 10 DF,  p-value: 0.3461

Analysis of Variance Table

Response: Score
             Df Sum Sq Mean Sq F value   Pr(>F)
FO(x1, x2)    2  14000    7000  1.0400 0.388732
TWI(x1, x2)   1    153     153  0.0227 0.883124
PQ(x1, x2)    2  28812   14406  2.1404 0.168362
Residuals    10  67305    6730
Lack of fit   6  65881   10980 30.8412 0.002611
Pure error    4   1424     356

Stationary point of response surface:
         x1          x2
 0.02585331 -0.06479614

Stationary point in original units:
         t       proc
0.04051707 2.93520386

Eigen analysis:
$values
[1]  -70.88796 -35.41911

$vectors
           [,1]       [,2]
[1,] -0.9998306  0.0184050
[2,] -0.0184050 -0.9998306
```

Tabla A5.6. *Salida del análisis de ajuste para el modelo de segundo orden, en caso del modelo evolutivo HKY85*

```
Call:
rsm(formula = Score ~ SO(x1, x2), data = current)

Residuals:
     Min       1Q   Median       3Q      Max
-201.340  -23.669    9.548   31.833  105.983

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 1121.152     33.545  33.422 1.36e-11 ***
x1             7.070     34.021   0.208   0.8396
x2            -5.366     33.545  -0.160   0.8761
x1:x2          8.851     27.778   0.319   0.7565
x1^2          74.840     45.714   1.637   0.1326
x2^2         -43.036     23.531  -1.829   0.0973 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 87.84 on 10 degrees of freedom
Multiple R-squared: 0.4363,   Adjusted R-squared: 0.1544
F-statistic: 1.548 on 5 and 10 DF,  p-value: 0.2599

Analysis of Variance Table

Response: Score
            Df Sum Sq Mean Sq  F value   Pr(>F)
FO(x1, x2)   2  21637   10819   1.4021   0.2906
TWI(x1, x2)  1    783     783   0.1015   0.7565
PQ(x1, x2)   2  37296   18648   2.4168   0.1392
Residuals   10  77160    7716
Lack of fit  6  76960   12827 255.8165 4.04e-05
Pure error   4    201      50

Stationary point of response surface:
         x1          x2
-0.04328032 -0.06679761

Stationary point in original units:
         t       proc
0.03913439 2.93320239

Eigen analysis:
$values
[1]  -75.00642 -43.20202

$vectors
            [,1]        [,2]
[1,] -0.99929788  0.03746653
[2,] -0.03746653 -0.99929788
```

Anexo 6. Evaluación de la exactitud de los modelos de Markov en la reconstrucción de la matriz de tasas de sustitución Q*, para la simulación en ausencia de la presión selectiva

Tabla 6.1. *Resultados de la evaluación para el modelo evolutivo TN93*

Variables independientes		**Variable respuesta**	
t	*proc*	ρ	*e*
0.02	1	0.6206	0.2998
0.04	1	0.4232	0.4017
0.06	1	0.5663	0.3483
0.02	2	0.2513	0.7073
0.04	2	0.4827	0.3701
0.06	2	0.5929	0.2782
0.02	3	0.4379	0.3867
0.06	3	0.4872	0.3691
0.02	4	0.6218	0.2047
0.04	4	0.5555	0.3559
0.06	4	0.7495	0.1851
0.04	3	0.6032	0.3221
0.04	3	0.9225	0.0488
0.04	3	0.5595	0.3547
0.04	3	0.4871	0.3698
0.04	3	0.3576	0.6450

***Tabla 6.2**. Resultados de la evaluación para el modelo evolutivo K80*

Variables independientes		**Variable respuesta**	
t	*proc*	ρ	*e*
0.02	1	0.5547	0.3113
0.04	1	0.5650	0.3209
0.06	1	0.4396	0.4178
0.02	2	0.6206	0.2179
0.04	2	0.7245	0.1855
0.06	2	0.4204	0.4574
0.02	3	0.6034	0.2571
0.06	3	0.4364	0.4220
0.02	4	0.3817	0.6302
0.04	4	0.4530	0.4081
0.06	4	0.5653	0.3206
0.04	3	0.5732	0.2998
0.04	3	0.6128	0.2378
0.04	3	0.6564	0.2029
0.04	3	0.4915	0.3671
0.04	3	0.3578	0.8237

***Tabla 6.3**. Resultados de la evaluación para el modelo evolutivo JC69*

Variables independientes		**Variable respuesta**	
t	*proc*	ρ	*e*
0.02	1	1	0
0.04	1	1	0
0.06	1	1	0
0.02	2	1	0
0.04	2	1	0
0.06	2	1	0
0.02	3	1	0
0.06	3	1	0
0.02	4	1	0
0.04	4	1	0
0.06	4	1	0
0.04	3	1	0
0.04	3	1	0
0.04	3	1	0
0.04	3	1	0
0.04	3	1	0

***Tabla 6.4**. Resultados de la evaluación para el modelo evolutivo SG09*

Variables independientes		**Variable respuesta**	
t	*proc*	ρ	*e*
0.02	1	0.4453	0.4267
0.04	1	0.7312	0.2458
0.06	1	0.3301	0.6189
0.02	2	0.2966	0.7416
0.04	2	0.5244	0.4022
0.06	2	0.8542	0.1338
0.02	3	0.1345	0.8425
0.06	3	0.9211	0.0628
0.02	4	0.3506	0.6021
0.04	4	0.7843	0.2140
0.06	4	0.5368	0.3831
0.04	3	0.4127	0.4563
0.04	3	0.7298	0.2869
0.04	3	0.6122	0.3612
0.04	3	0.2457	0.7389
0.04	3	0.6386	0.3103

***Tabla 6.5**. Resultados de la evaluación para el modelo evolutivo F81*

Variables independientes		**Variable respuesta**	
t	*proc*	ρ	*e*
0.02	1	0.5826	0.4048
0.04	1	0.9908	0.1012
0.06	1	0.4553	0.4602
0.02	2	0.3175	0.6483
0.04	2	0.3057	0.6747
0.06	2	0.1526	0.9809
0.02	3	0.2145	0.8430
0.06	3	0.2433	0.8179
0.02	4	0.8352	0.2386
0.04	4	0.9653	0.0598
0.06	4	0.2105	0.8538
0.04	3	0.3763	0.6185
0.04	3	0.1662	0.8587
0.04	3	0.1325	0.8814
0.04	3	0.1895	0.7802
0.04	3	0.1987	0.7133

***Tabla 6.6**. Resultados de la evaluación para el modelo evolutivo HKY85*

Variables independientes		**Variable respuesta**	
t	*proc*	ρ	*e*
0.02	1	0.9279	0.0566
0.04	1	0.8765	0.1819
0.06	1	0.9767	0.0267
0.02	2	0.5506	0.4396
0.04	2	0.5583	0.4307
0.06	2	0.9760	0.0278
0.02	3	0.2143	0.8744
0.06	3	0.7743	0.2805
0.02	4	0.8148	0.2584
0.04	4	0.9124	0.1570
0.06	4	0.9520	0.0617
0.04	3	0.4662	0.5618
0.04	3	0.4757	0.4840
0.04	3	0.3736	0.6235
0.04	3	0.4823	0.4353
0.04	3	0.3810	0.5893

Anexo 7. Diseños experimentales de la simulación bajo la presión selectiva

Tabla A7.1. *Diseño experimental correspondiente al modelo evolutivo TN93*

Variables naturales		**Variables codificadas**		**Respuesta**
t	*proc*	x_1	x_2	*Alignment Score (% de similitud)*
0.02	1	-1	-2	819.53 (72%)
0.04	1	0	-2	899.40 (78%)
0.06	1	1	-2	963.13 (83%)
0.02	2	-1	-1	1151.93 (97%)
0.04	2	0	-1	1146.80 (97%)
0.06	2	1	-1	1142.33 (96%)
0.02	3	-1	0	1142.73 (96%)
0.06	3	1	0	1129.13 (95%)
0.02	4	-1	1	1135.53 (96%)
0.04	4	0	1	1132.20 (95%)
0.06	4	1	1	1133.80 (96%)
0.04	3	0	0	1134.36 (96%)
0.04	3	0	0	1140.13 (96%)
0.04	3	0	0	1144.13 (96%)
0.04	3	0	0	1134.87 (96%)
0.04	3	0	0	1141.87 (96%)

Tabla A7.2. *Diseño experimental correspondiente al modelo evolutivo K80*

Variables naturales		**Variables codificadas**		**Respuesta**
t	*proc*	x_1	x_2	*Alignment Score (% de similitud)*
0.02	1	-1	-2	1029.67 (88%)
0.04	1	0	-2	1050.27 (89%)
0.06	1	1	-2	1086.33 (91%)
0.02	2	-1	-1	1148.40 (97%)
0.04	2	0	-1	1146.13 (96%)
0.06	2	1	-1	1147.53 (97%)
0.02	3	-1	0	1155.33 (97%)
0.06	3	1	0	1145.93 (97%)
0.02	4	-1	1	1142.13 (96%)
0.04	4	0	1	1145.20 (96%)
0.06	4	1	1	1140.57 (96%)
0.04	3	0	0	1138.80 (96%)
0.04	3	0	0	1141.80 (96%)
0.04	3	0	0	1147.53 (97%)
0.04	3	0	0	1154.60 (97%)
0.04	3	0	0	1149.47 (97%)

Tabla A7.3. *Diseño experimental correspondiente al modelo evolutivo JC69*

Variables naturales		**Variables codificadas**		**Respuesta**
t	*proc*	x_1	x_2	*Alignment Score (% de similitud)*
0.02	1	-1	-2	977.40 (84%)
0.04	1	0	-2	1138.53 (96%)
0.06	1	1	-2	1147.93 (97%)
0.02	2	-1	-1	1143.40 (96%)
0.04	2	0	-1	1145.53 (97%)
0.06	2	1	-1	1124.07 (95%)
0.02	3	-1	0	1148.27 (97%)
0.06	3	1	0	1134.00 (96%)
0.02	4	-1	1	1156.73 (97%)
0.04	4	0	1	1146.87 (97%)
0.06	4	1	1	1148.60 (97%)
0.04	3	0	0	1132.07 (96%)
0.04	3	0	0	1138.67 (96%)
0.04	3	0	0	1121.13 (95%)
0.04	3	0	0	1147.40 (97%)
0.04	3	0	0	1140.33 (96%)

Tabla A7.4. *Diseño experimental correspondiente al modelo evolutivo SG09*

Variables naturales		**Variables codificadas**		**Respuesta**
t	*proc*	x_1	x_2	*Alignment Score (% de similitud)*
0.02	1	-1	-2	1153.20 (97%)
0.04	1	0	-2	1038.17 (94%)
0.06	1	1	-2	955.73 (83%)
0.02	2	-1	-1	1155.20 (97%)
0.04	2	0	-1	1145.87 (96%)
0.06	2	1	-1	1146.20 (97%)
0.02	3	-1	0	1147.13 (97%)
0.06	3	1	0	1130.07 (95%)
0.02	4	-1	1	1144.93 (96%)
0.04	4	0	1	1140.80 (96%)
0.06	4	1	1	1142.87 (96%)
0.04	3	0	0	1144.27 (97%)
0.04	3	0	0	1152.13 (97%)
0.04	3	0	0	1149.00 (97%)
0.04	3	0	0	1141.20 (96%)
0.04	3	0	0	1150.27 (97%)

Tabla A7.5. *Diseño experimental correspondiente al modelo evolutivo F81*

Variables naturales		**Variables codificadas**		**Respuesta**
t	*proc*	x_1	x_2	*Alignment Score (% de similitud)*
0.02	1	-1	-2	1025.07 (87%)
0.04	1	0	-2	1036.80 (88%)
0.06	1	1	-2	1063.00 (91%)
0.02	2	-1	-1	1160.73 (98%)
0.04	2	0	-1	1140.73 (96%)
0.06	2	1	-1	1131.53 (95%)
0.02	3	-1	0	1137.07 (96%)
0.06	3	1	0	1124.67 (95%)
0.02	4	-1	1	1129.73 (95%)
0.04	4	0	1	1139.07 (96%)
0.06	4	1	1	1137.53 (98%)
0.04	3	0	0	1133.60 (96%)
0.04	3	0	0	1137.40 (96%)
0.04	3	0	0	1143.13 (96%)
0.04	3	0	0	1144.53 (96%)
0.04	3	0	0	1143.80 (96%)

Tabla A7.6. *Diseño experimental correspondiente al modelo evolutivo HKY85*

Variables naturales		**Variables codificadas**		**Respuesta**
t	*proc*	x_1	x_2	*Alignment Score (% de similitud)*
0.02	1	-1	-2	1150.27 (97%)
0.04	1	0	-2	1138.93 (96%)
0.06	1	1	-2	1125.07 (95%)
0.02	2	-1	-1	1133.93 (95%)
0.04	2	0	-1	1138.00 (96%)
0.06	2	1	-1	1121.47 (95%)
0.02	3	-1	0	1128.80 (95%)
0.06	3	1	0	1134.87 (96%)
0.02	4	-1	1	1145.80 (96%)
0.04	4	0	1	1146.00 (97%)
0.06	4	1	1	1145.73 (96%)
0.04	3	0	0	1127.80 (95%)
0.04	3	0	0	1130.60 (95%)
0.04	3	0	0	1132.60 (96%)
0.04	3	0	0	1144.33 (96%)
0.04	3	0	0	1148.00 (97%)

Anexo 8. Resultados obtenidos de los diseños experimentales RSM realizados en R para la simulación bajo la presión selectiva

Tabla A8.1. *Salida del análisis de ajuste para el modelo de segundo orden, en caso del modelo evolutivo TN93*

```
Residuals:
    Min      1Q  Median      3Q     Max
-48.298 -20.913  -8.008  17.526  76.360

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 1155.656     15.993  72.261 6.28e-15 ***
x1             3.834     16.220   0.236 0.817909
x2            10.226     15.993   0.639 0.536926
x1:x2        -22.000     13.243  -1.661 0.127660
x1^2          14.479     21.794   0.664 0.521481
x2^2         -58.506     11.218  -5.215 0.000393 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 41.88 on 10 degrees of freedom
Multiple R-squared: 0.8886,   Adjusted R-squared: 0.8329
F-statistic: 15.95 on 5 and 10 DF,  p-value: 0.0001741

Analysis of Variance Table

Response: Score
             Df Sum Sq Mean Sq  F value    Pr(>F)
FO(x1, x2)    2  86578   43289  24.6824 0.0001356
TWI(x1, x2)   1   4840    4840   2.7595 0.1276596
PQ(x1, x2)    2  48453   24226  13.8133 0.0013259
Residuals    10  17538    1754
Lack of fit   6  17464    2911 156.5057 0.107063
Pure error    4     74      19

Stationary point of response surface:
         x1          x2
-0.05775471  0.09825168

Stationary point in original units:
         t       proc
0.03884491 3.09825168

Eigen analysis:
$values
[1]  -16.10121 -60.12736

$vectors
           [,1]      [,2]
[1,] -0.9893053 0.1458595
[2,]  0.1458595 0.9893053
```

Tabla A8.2. *Salida del análisis de ajuste para el modelo de segundo orden, en caso del modelo evolutivo K80*

```
Call:
rsm(formula = Score ~ SO(x1, x2), data = current)

Residuals:
    Min      1Q  Median      3Q     Max
-15.159  -7.150  -3.127   4.911  24.127

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 1152.014      5.483 210.117  < 2e-16 ***
x1             1.024      5.560   0.184 0.857572
x2             2.833      5.483   0.517 0.616549
x1:x2         -9.160      4.540  -2.017 0.071283 .
x1^2           8.050      7.472   1.077 0.306582
x2^2         -22.774      3.846  -5.922 0.000147 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 14.36 on 10 degrees of freedom
Multiple R-squared: 0.9068,   Adjusted R-squared: 0.8602
F-statistic: 19.47 on 5 and 10 DF,  p-value: 7.297e-05

Analysis of Variance Table

Response: Score
             Df  Sum Sq Mean Sq F value    Pr(>F)
FO(x1, x2)    2 11960.0  5980.0 29.0112 6.867e-05
TWI(x1, x2)   1   839.0   839.0  4.0701  0.071283
PQ(x1, x2)    2  7262.7  3631.4 17.6170  0.000528
Residuals    10  2061.3   206.1
Lack of fit   6  1904.4   317.4  8.0943  0.131319
Pure error    4   156.9    39.2

Stationary point of response surface:
          x1          x2
-0.02531612  0.06729509

Stationary point in original units:
          t       proc
0.03949368 3.06729509

Eigen analysis:
$values
[1]   -8.716454 -23.440482

$vectors
           [,1]      [,2]
[1,] -0.9895898 0.1439169
[2,]  0.1439169 0.9895898
```

Tabla A8.3. *Salida del análisis de ajuste para el modelo de segundo orden, en caso del modelo evolutivo JC69*

```
Call:
rsm(formula = Score ~ SO(x1, x2), data = current)

Residuals:
    Min      1Q  Median      3Q     Max
-56.657 -10.934  -5.366  14.414  52.898

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 1146.291     12.465  91.959 5.66e-16 ***
x1             2.827     12.642   0.224   0.8276
x2            11.487     12.465   0.922   0.3785
x1:x2        -26.546     10.322  -2.572   0.0278 *
x1^2          -8.792     16.987  -0.518   0.6160
x2^2          -6.137      8.744  -0.702   0.4988
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 32.64 on 10 degrees of freedom
Multiple R-squared: 0.5963,   Adjusted R-squared: 0.3944
F-statistic: 2.954 on 5 and 10 DF,  p-value: 0.06819

Analysis of Variance Table

Response: Score
             Df  Sum Sq Mean Sq F value   Pr(>F)
FO(x1, x2)    2  7620.8  3810.4  3.5762 0.067355
TWI(x1, x2)   1  7046.9  7046.9  6.6138 0.027808
PQ(x1, x2)    2  1069.4   534.7  0.5018 0.619883
Residuals    10 10654.8  1065.5
Lack of fit   6 10262.4  1710.4 17.4368 0.077143
Pure error    4   392.4    98.1

Stationary point of response surface:
        x1         x2
 0.9527772 -0.0596470

Stationary point in original units:
         t       proc
0.05905554 2.94035298

Eigen analysis:
$values
[1]   -5.874792 -20.803588

$vectors
           [,1]       [,2]
[1,]  0.6710081 -0.7414500
[2,] -0.7414500 -0.6710081
```

Tabla A8.4. *Salida del análisis de ajuste para el modelo de segundo orden, en caso del modelo evolutivo SG09*

```
Call:
rsm(formula = Score ~ SO(x1, x2), data = current)

Residuals:
    Min      1Q  Median      3Q     Max
-30.868 -13.007  -4.512  12.519  56.961

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 1151.147     10.411 110.570  < 2e-16 ***
x1           -13.745     10.559  -1.302  0.22220
x2             3.469     10.411   0.333  0.74583
x1:x2         28.908      8.621   3.353  0.00733 **
x1^2           7.636     14.188   0.538  0.60222
x2^2         -23.421      7.303  -3.207  0.00938 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 27.26 on 10 degrees of freedom
Multiple R-squared: 0.8262,   Adjusted R-squared: 0.7393
F-statistic: 9.508 on 5 and 10 DF,  p-value: 0.001472

Analysis of Variance Table

Response: Score
             Df  Sum Sq Mean Sq F value    Pr(>F)
FO(x1, x2)    2 19276.9  9638.4 12.9681 0.0016685
TWI(x1, x2)   1  8357.0  8357.0 11.2440 0.0073256
PQ(x1, x2)    2  7698.1  3849.0  5.1787 0.0286014
Residuals    10  7432.4   743.2
Lack of fit   6  7351.0  1225.2 60.2023 0.709423
Pure error    4    81.4    20.4

Stationary point of response surface:
       x1        x2
0.0350427 0.1903310

Stationary point in original units:
         t       proc
0.04070855 3.19033100

Eigen analysis:
$values
[1]  -13.32178 -29.10720

$vectors
           [,1]       [,2]
[1,] -0.9305827 -0.3660817
[2,] -0.3660817  0.9305827
```

Tabla A8.5. *Salida del análisis de ajuste para el modelo de segundo orden, en caso del modelo evolutivo F81*

```
Call:
rsm(formula = Score ~ SO(x1, x2), data = current)

Residuals:
    Min      1Q  Median      3Q     Max
-43.733 -10.642  -2.414   3.415  50.510

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 1146.5794     9.6670 118.607   <2e-16 ***
x1             2.3265     9.8041   0.237   0.8172
x2             0.1197     9.6670   0.012   0.9904
x1:x2        -14.6295     8.0050  -1.828   0.0976 .
x1^2         -10.3943    13.1739  -0.789   0.4484
x2^2          -8.8894     6.7811  -1.311   0.2192
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 25.31 on 10 degrees of freedom
Multiple R-squared: 0.5065,  Adjusted R-squared: 0.2597
F-statistic: 2.053 on 5 and 10 DF,  p-value: 0.1559

Analysis of Variance Table

Response: Score
            Df Sum Sq Mean Sq F value   Pr(>F)
FO(x1, x2)   2 2485.2  1242.6  1.9391 0.194231
TWI(x1, x2)  1 2140.2  2140.2  3.3399 0.097568
PQ(x1, x2)   2 1951.3   975.7  1.5225 0.264706
Residuals   10 6408.1   640.8
Lack of fit  6 6316.8  1052.8 46.1414 0.001194
Pure error   4   91.3    22.8

Stationary point of response surface:
        x1         x2
 0.2546127 -0.2027782

Stationary point in original units:
         t       proc
0.04509225 2.79722177

Eigen analysis:
$values
[1]  -2.288476 -16.995181

$vectors
           [,1]       [,2]
[1,]  0.6699511 -0.7424053
[2,] -0.7424053 -0.6699511
```

Tabla A8.6. *Salida del análisis de ajuste para el modelo de segundo orden, en caso del modelo evolutivo HKY85*

```
Call:
rsm(formula = Score ~ SO(x1, x2), data = current)

Residuals:
   Min     1Q Median     3Q    Max
-8.730 -3.001 -1.821  4.095 11.470

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 1136.530      2.500 454.665   <2e-16 ***
x1            -1.610      2.535  -0.635   0.5398
x2             7.369      2.500   2.948   0.0146 *
x1:x2          4.696      2.070   2.269   0.0467 *
x1^2          -4.293      3.407  -1.260   0.2362
x2^2           4.793      1.753   2.734   0.0211 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.546 on 10 degrees of freedom
Multiple R-squared: 0.6429,  Adjusted R-squared: 0.4643
F-statistic: 3.601 on 5 and 10 DF,  p-value: 0.04018

Analysis of Variance Table

Response: Score
             Df Sum Sq Mean Sq F value  Pr(>F)
FO(x1, x2)    2 215.37  107.68  2.5132 0.13053
TWI(x1, x2)   1 220.52  220.52  5.1468 0.04668
PQ(x1, x2)    2 335.50  167.75  3.9151 0.05549
Residuals    10 428.47   42.85
Lack of fit   6 109.34   18.22  0.2284 0.94625
Pure error    4 319.13   79.78

Stationary point of response surface:
         x1          x2
-0.0155682 0.06620694

Stationary point in original units:
          t        proc
0.04031136 3.06620694

Eigen analysis:
$values
[1]  -5.364138 -4.863675

$vectors
           [,1]       [,2]
[1,] 0.2362585 -0.9716902
[2,] 0.9716902  0.2362585
```

Anexo 9. Evaluación de la exactitud de los modelos de Markov en la reconstrucción de la matriz de tasas de sustitución Q^* para la simulación bajo la presión selectiva

Tabla 9.1. *Resultados de la evaluación para el modelo evolutivo TN93*

Variables independientes		**Variable respuesta**	
t	*proc*	ρ	*e*
0.02	1	0.3111	0.7299
0.04	1	0.4060	0.5323
0.06	1	0.2062	0.9097
0.02	2	0.5965	0.4834
0.04	2	0.5488	0.5069
0.06	2	0.8302	0.2129
0.02	3	0.6175	0.3885
0.06	3	0.8513	0.1709
0.02	4	0.9677	0.0137
0.04	4	0.4699	0.4979
0.06	4	0.5086	0.5721
0.04	3	0.6423	0.3169
0.04	3	0.8528	0.1561
0.04	3	0.9722	0.0119
0.04	3	0.7945	0.2975
0.04	3	0.6236	0.3299

***Tabla 9.2**. Resultados de la evaluación para el modelo evolutivo K80*

Variables independientes		**Variable respuesta**	
t	*proc*	ρ	*e*
0.02	1	0.3971	0.7132
0.04	1	0.7949	0.2796
0.06	1	0.4286	0.5213
0.02	2	0.6199	0.3162
0.04	2	0.1919	0.8669
0.06	2	0.5835	0.3766
0.02	3	0.9737	0.0290
0.06	3	0.2850	0.7529
0.02	4	0.8033	0.2631
0.04	4	0.3094	0.6948
0.06	4	0.3978	0.6112
0.04	3	0.9368	0.0528
0.04	3	0.9029	0.0755
0.04	3	0.9140	0.0725
0.04	3	0.9578	0.0421
0.04	3	0.8227	0.1054

***Tabla 9.3**. Resultados de la evaluación para el modelo evolutivo JC69*

Variables independientes		**Variable respuesta**	
t	*proc*	ρ	*e*
0.02	1	1	0
0.04	1	1	0
0.06	1	1	0
0.02	2	1	0
0.04	2	1	0
0.06	2	1	0
0.02	3	1	0
0.06	3	1	0
0.02	4	1	0
0.04	4	1	0
0.06	4	1	0
0.04	3	1	0
0.04	3	1	0
0.04	3	1	0
0.04	3	1	0
0.04	3	1	0

***Tabla 9.4**. Resultados de la evaluación para el modelo evolutivo SG09*

Variables independientes		**Variable respuesta**	
t	*proc*	ρ	*e*
0.02	1	0.5664	0.4021
0.04	1	0.8471	0.1146
0.06	1	0.4903	0.5725
0.02	2	0.4874	0.5994
0.04	2	0.7917	0.2410
0.06	2	0.2797	0.6859
0.02	3	0.6061	0.2933
0.06	3	0.7882	0.2669
0.02	4	0.3784	0.6531
0.04	4	0.8976	0.0878
0.06	4	0.3701	0.6352
0.04	3	0.9821	0.0251
0.04	3	0.8707	0.1040
0.04	3	0.8094	0.1949
0.04	3	0.8115	0.1920
0.04	3	0.9153	0.0954

***Tabla 9.5**. Resultados de la evaluación para el modelo evolutivo F81*

Variables independientes		**Variable respuesta**	
t	*proc*	ρ	*e*
0.02	1	0.2626	0.7891
0.04	1	0.4500	0.5380
0.06	1	0.7218	0.2311
0.02	2	0.5060	0.4997
0.04	2	0.2237	0.8269
0.06	2	0.3682	0.5999
0.02	3	0.2565	0.8075
0.06	3	0.9975	0.0118
0.02	4	0.3042	0.7316
0.04	4	0.6241	0.3634
0.06	4	0.2483	0.8148
0.04	3	0.8200	0.1833
0.04	3	0.8986	0.0719
0.04	3	0.7196	0.2724
0.04	3	0.6632	0.2998
0.04	3	0.7752	0.2178

***Tabla 9.6**. Resultados de la evaluación para el modelo evolutivo HKY85*

Variables independientes		**Variable respuesta**	
t	*proc*	ρ	*e*
0.02	1	0.5965	0.4186
0.04	1	0.3611	0.7662
0.06	1	0.9791	0.0462
0.02	2	0.7981	0.2210
0.04	2	0.4452	0.4265
0.06	2	0.1672	0.8004
0.02	3	0.2562	0.8733
0.06	3	0.9233	0.0757
0.02	4	0.4422	0.4991
0.04	4	0.5653	0.3554
0.06	4	0.9036	0.1057
0.04	3	0.9761	0.0483
0.04	3	0.9798	0.0427
0.04	3	0.9813	0.0314
0.04	3	0.9825	0.0217
0.04	3	0.8934	0.0585

Anexo 10. Resumen de los resultados obtenidos en dos casos de simulación

Tabla 10.1. *Valores óptimos de la Puntuación de Alineamiento (Alignment Score) y el error relativo (e) de la estimación de la matriz de tasas para diferentes modelos de Markov*

	Modelos de Markov	***t***	***proc***	***Score***	***e***
Simulación sin restricciones	TN93	0.04	3	1153.2017	0.0488
	K80	0.04	3	1128.8551	0.2029
	JC69	0.02	4	1142.2201	0
	SG09	0.04	3	1141.9432	0.2869
	F81	0.04	3	1119.1558	0.6185
	HKY85	0.04	3	1121.1782	0.4353
Simulación bajo la presión selectiva	TN93	0.04	3	1156.0476	0.0119
	K80	0.04	3	1152.0963	0.0421
	JC69	0.06	3	1145.5809	0
	SG09	0.04	3	1149.2421	0.0251
	F81	0.04	3	1146.8634	0.0719
	HKY85	0.04	3	1134.9491	0.2210