Implementación del fitness dependiente de la frecuencia genética en el paquete OncoSimulR

Sergio Sánchez Carrillo

Máster en Bioinformática y Biología Computacional (ByBC)

MÁSTERES DE LA UAM 2017 - 2018

Escuela Politécnica Superior

UAM Universidad Autónoma de Madrid

Universidad Autónoma de Madrid
Escuela politécnica superior

Máster Oficial en Bioinformática y Biología Computacional (ByBC)

Trabajo Fin de Máster

IMPLEMENTACIÓN DEL FITNESS DEPENDIENTE DE LA FRECUENCIA GENÉTICA EN EL PAQUETE OncoSimulR

Autor: Sergio Sánchez Carrillo
Director: Ramón Díaz Uriarte
Septiembre de 2018

IMPLEMENTACIÓN DEL FITNESS DEPENDIENTE DE LA FRECUENCIA GENÉTICA EN EL PAQUETE OncoSimulR

Autor: Sergio Sánchez Carrillo
Director: Ramón Díaz Uriarte

Departamento de Bioquímica
Universidad Autónoma de Madrid

Septiembre de 2018

Resumen

Resumen

En el estudio del cáncer es fundamental conocer las interacciones entre las distintas subpoblaciones celulares que componen el tumor. Para ello, son determinantes los modelos de progresión tumoral, gracias a los cuales podemos entender y explorar los factores relevantes en la evolución de una población de celulas tumorales, con lo que es posible conocer mejor los distintos escenarios en los combatir el cáncer. En este contexto, en el que pretendemos modelar la evolución tumoral teniendo en cuenta las interacciones entre los diferentes subclones, destacan los escenarios en los que el fitness de cada subpoblación depende de la frecuencia genotípica. Entre los simuladores disponibles hasta la fecha es de resaltar el paquete de R/Bioconductor OncoSimulR, que hace especial hincapié en las relaciones de epistasia entre genes y en el orden de aparición de las mutaciones, siendo remarcable, entre otras características, la elevada flexibilidad de este en la especificación de las interacciones entre los genes. Hasta la fecha, este software permitía la definición del fitness de cada clon como una función arbitraria de las interacciones genéticas entre múltiples genes o módulos de estos, con un fitness landscape fijo e incluso como dependiente de la densidad poblacional, pero no era posible la especificación explícita del fitness dependiente de la frecuencia genética. Por ello, este trabajo aborda la implementación de la posibilidad de realizar las simulaciones de modo que el fitness dependa de la frecuencia genética, funcionalidad crucial para permitir ampliar las situaciones en las que el paquete es capaz de servir de herramienta predictiva y de análisis, teniendo en cuenta las interacciones entre los distintos clones en la dinámica evolutiva de la progresión tumoral. De este modo, se permite con mayor eficacia la aplicabilidad de los modelos a problemas reales, entre los que se destacan la búsqueda de dianas terapéuticas.

Palabras Clave

Cáncer, fitness landscape, modelos de progresión tumoral, R, seleción dependiente de la frecuencia.

Abstract

In the study of cancer it is essential to know the interactions between the different cellular subpopulations that make up the tumor. To this end, the models of tumor progression are decisive, thanks to which we can understand and explore the relevant factors in the evolution of a population of tumor cells, with which it is possible to better understand the possible scenarios in the fight against cancer. In this context, in which we intend to model tumor evolution taking into account the interactions between the different subclones, we highlight the scenarios where the fitness of each subpopulation depends on the genotypic frequency. Among the simulators available to date, it is worth highlighting the R / Bioconductor OncoSimulR package, which places special emphasis on the relationship of epistasia between genes and the order of appearance of mutations, being remarkable, among other characteristics, the high flexibility of this software in the specification of the interactions between genes. Until now, this software allowed the definition of the fitness of each clone as an arbitrary function of the genetic interactions between multiple genes or modules of these, with a fixed fitness landscape and even as dependent on population density, but the specification was not possible explicit fitness dependent on genetic frequency. Therefore, this work addresses the implementation of the possibility of performing the simulations so that the fitness depends on the genetic frequency, crucial functionality to allow to expand the situations in which the package is able to serve as a predictive and analytical tool, taking into account the interactions between different clones in the evolutionary dynamics of tumor progression. In this way, the applicability of the models to real problems is more effectively allowed, among which the search for therapeutic targets stands out.

Key words

Cancer, fitness landscape, frequency dependent selection, R, tumor progression models.

Agradecimientos

Me gustaría aprovechar la ocasión para agradecer el apoyo y la paciencia recibida por parte del director de ese trabajo, ya que me decanté a realizar un Trabajo Fin de Máster de desarrollo de software siendo biólogo de formación, y esto, me ha llevado a aprender mucho por un lado, pero también, por otro, a necesitar de gran ayuda .

También quiero agradecer a mi familia, a los están y a los que se fueron, toda la ayuda que me han brindado en la extravagancia de ponerme a estudiar rondando los cuarenta.

Y gracias a ti Ainara, que me has apoyado tanto todos estos años de estudio, que son los que hace que nos conocemos. Sin tu cariño nada de esto hubiera sido posible. ¡GRACIAS!

Índice general

Índice de Figuras

Índice de Tablas

Índice de Code Boxes

1 Introducción

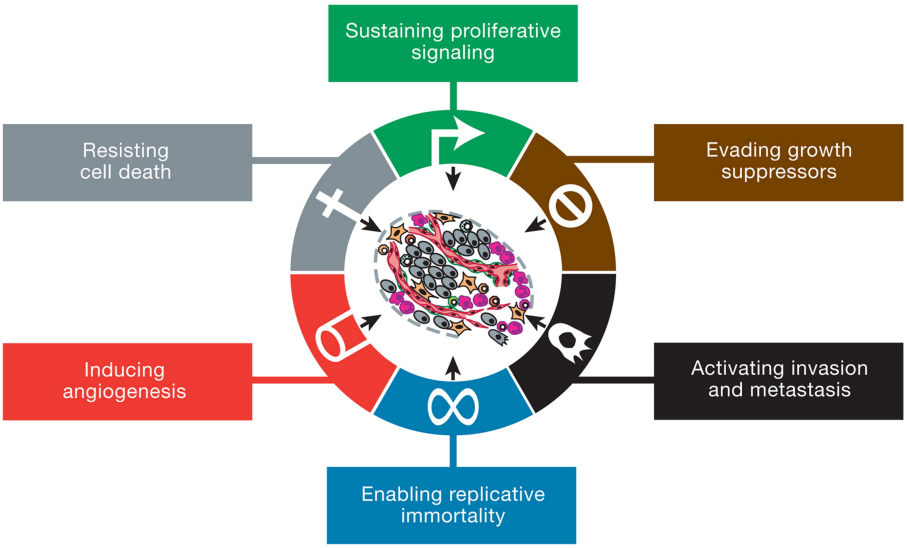
1.1. Motivación del Proyecto

El cáncer es una enfermedad de etiología compleja, que se produce cuando en un tejido acontecen los fenómenos que Hanahan and Weinberg (2000) denominaron sello de identidad del cáncer (Hallmarks of Cancer). Este sello de identidad del cáncer se manifiesta en una población celular cuando ocurren los siguientes acontecimientos (Figura 1.1.a):

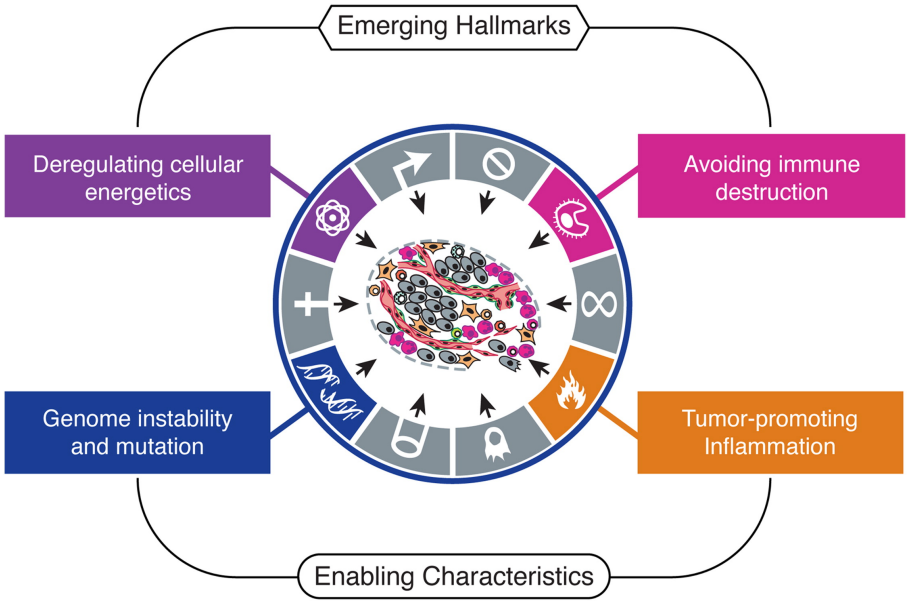
- Evasión de la apoptosis, o muerte celular programada
- Las células de la población son capaces de producir sus propias señales proliferativas, no dependiendo de las externas
- La población celular se insensibiliza a las señales encaminadas a interrumpir su crecimiento
- Aparece una capacidad proliferativa ilimitada
- Se desarrolla el fenómeno de angiogénesis, es decir se generan nuevos vasos sanguíneos que irrigan el tumor
- Se produce la metástasis, o invasión de otros tejidos distintos del originario

A estas seis marcas de identidad del cáncer, Hanahan and Weinberg (2011) en base a los nuevos avances en biología del cáncer, añadieron otras dos marcas que denominaron emergentes (Figura 1.1.b), debido a que estas parecen comportarse como las propiedades de idéntico nombre que aparecen en los sistemas complejos:

- Desregulación del metabolismo energético celular, que permite soportar con mayor eficiencia la proliferación neoplásica
- Evasión de la respuesta inmune, de modo que el tumor es capaz de esquivar la destrucción por parte de las células del sistema inmunitario



(a)



(b)

Figura 1.1: a) Marcas de identidad del cáncer. b) Marcas emergentes y características habilitantes. Figuras tomadas de Hanahan and Weinberg (2011).

Todos estos fenómenos producen un crecimiento fuera de control de la población celular neoplásica, cuyo origen descansa en las que se han denominado características habilitantes (Hanahan and Weinberg, 2011) (Figura 1.1.b). Estas son la inestabilidad genómica y la respuesta inflamatoria. La primera hace referencia a que las marcas son el resultado en gran medida de alteraciones tanto genéticas como cromosómicas del genoma celular. La segunda señala que los tumores están fuertemente inducidos por la respuesta inmune inflamatoria producida por las células del sistema inmunitario, ya que esta parece favorecer el aporte de moléculas bioactivas (factores de crecimiento, factores limitadores de la muerte celular, enzimas que favorecen la angiogénesis, etc) que ayudan a la aparición de las marcas del cáncer.

El punto de vista evolutivo en la biología del cáncer considera la proliferación tumoral como sujeta a la dinámica darwiniana, en la que no sólo la aparición de mutaciones aleatorias en genes drivers (directamente relacionados con la tumorigénesis) son las responsables del cáncer, si no

que también tienen una importancia primordial las fuerzas selectivas derivadas del ambiente (Korolev et al., 2014; Lloyd et al., 2016). Esta perspectiva enmarca el desarrollo de la población tumoral en el paradigma darwiniano de la lucha por la supervivencia, concibiéndose este como un proceso más de índole evolutiva y ecológica, al igual que todos aquellos relacionados con el fenómeno biológico (Merlo et al., 2006; Greaves and Maley, 2012; Gerlinger et al., 2012). De este modo, la metodología propia de la Biología Evolutiva es aplicable al estudio de la progresión de las poblaciones tumorales, donde puede aportar interesantes puntos de vista, como por ejemplo en el desarrollo de fármacos (Pepper et al., 2009).

Esta perpectiva evolutiva en la biología del cáncer es hoy en día ampliamente aceptada, concibiéndose el cáncer como una enfermedad causada por evolución clonal (Nowell, 1976), es decir, la progresión tumoral consiste en la aparición de distintos clones que van adquiriendo las ya comentadas marcas del cáncer (Figura 1.2.a). Algunas de estas marcas, como la evasión de la apoptosis, pueden beneficiar sólo a linajes individuales, mientras que otras, como la angiogénesis, pueden beneficiar a la totalidad del tumor (Korolev et al., 2014). De esta manera, en el paradigma de la evolución clonal podemos hablar de dos modelos, en principio contrapuestos, la interferencia clonal (Figura 1.2.b), en la que los distintos linajes compiten entre ellos, y la cooperación clonal (Figura 1.2.c), que acontece cuando los clones desarrollan una relación mutualista.

La cooperación clonal en el desarrollo tumoral fue discutida por Axelrod et al. (2006) en un trabajo fundacional que proponía que la progresión desde un estado neoplásico a la malignidad podría ser vista como un proceso de cooperación de tipo mutualista entre los distintos clones que componen el tumor, que colaboran entre sí a través de la transmisión de productos difusibles, como son los factores de crecimiento. De este modo, el microambiente tumoral adquiere un papel fundamental en el desarrollo de la enfermedad.

Evidencias de la cooperación clonal en el desarrollo tumoral a través de productos difusibles han sido encontradas por Cleary et al. (2014) y por Marusyk et al. (2014) en cáncer de pecho en modelo murino. Por otro lado, también en ratón, Ohtsuka et al. (2018) ha demostrado recientemente en cáncer de estómago la existencia de estas relaciones de cooperación clonal.

Dicha cooperación clonal es un fenómeno que puede ser entendido en el marco de la selección dependiente de la frecuencia (frequency dependent selection), que en este contexto establece que el fitness de un clon va a depender de las abundancias relativas de los distintos clones. Esto es debido a que los efectos cooperativos van a depender de las frecuencias relativas de los distintos actores de la dinámica evolutiva.

En el estudio de los fenómenos de cooperación clonal en el marco de la progresión tumoral son fundamentales los experimentos de simulación debido a que con su empleo se pueden verificar resultados analíticos, generar datos para estudiar el comportamiento de métodos estadísticos, así como examinar modelos matemáticamente intratables (Diaz-Uriarte, 2017). Existen en la actualidad diferentes simuladores disponibles (véase el capítulo Modelos de Progresión Tumoral) para realizar estas simulaciones, entre los cuales destacamos el paquete de R OncoSimulR (Diaz-Uriarte, 2017), debido a que con el es posible realizar de una manera sencilla y flexible la especificación del fitness y de los efectos mutacionales.

Hasta la fecha, en OncoSimulR (Diaz-Uriarte, 2017) no era posible emplear los modelos de progresión tumoral disponibles de modo que el fitness fuera dependiente de la frecuencia genotípica, lo cual, como venimos argumentando es fundamental si queremos simular un rango amplio de circunstancias, resaltando entre ellas el caso de la cooperación clonal.

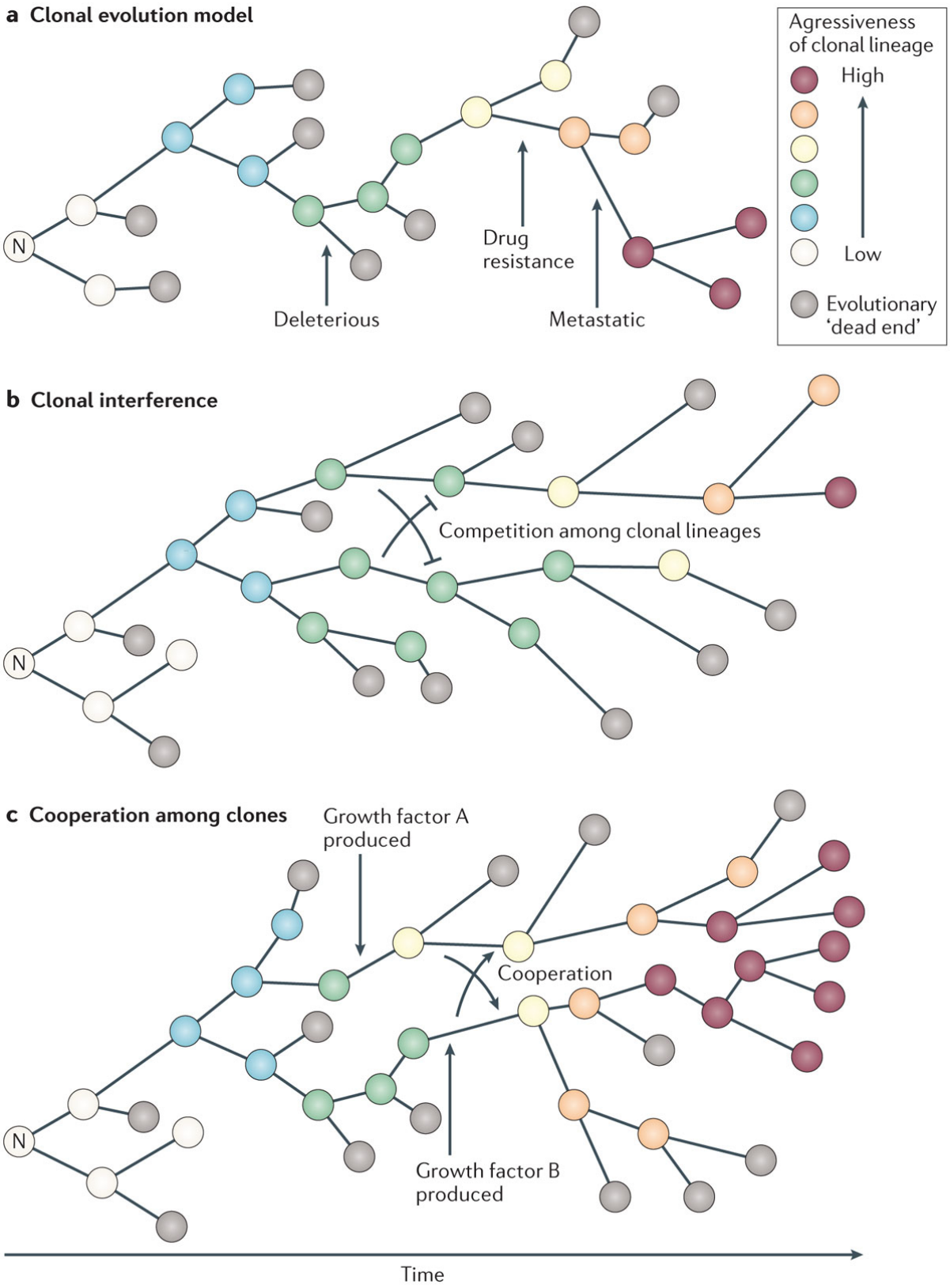


Figura 1.2: a) Una neoplasia benigna en principio acumula mutaciones apareciendo fenotipos cancerosos de mayor agresividad y mayor fitness que compiten y esplazan a los demás. b)Distintos clones con un fitness similar coexisten y compiten. c) Clones con características complementarias establecen una relación mutualista y fruto de la cooperación aparece la malignidad del tumor. Figura tomada de Korolev et al. (2014).

1.2. Objetivos

A pesar de que OncoSimulR (Diaz-Uriarte, 2017) ya es un software muy eficaz y flexible como simulador de la progresión tumoral, la inclusión de la funcionalidad de emplear los modelos disponibles con el fitness dependiente de la frecuencia genotípica es fundamental para hacer las simulaciones lo más realistas posibles, así como ampliar los distintos escenarios que el paquete será capaz de modelar. Con el fitness dependiente de la frecuencia genotípica, el fitness landscape es un paisaje dinámico en el que el fitness de cada clon es dependiente de la frecuencia genotípica de todas las subpoblaciones clonales dentro de la población tumoral. Todo ello, permitirá de un modo más eficaz la aplicabilidad de los modelos a problemas reales, entre los que destacamos la búsqueda de dianas terapéuticas.

Por lo dicho con anterioridad, se planteó el principal objetivo de este Trabajo Fin de Máster, que como indica su título, no es otro más que la implementación de la funcionalidad de fitness dependiente de la frecuencia genética en el paquete OncoSimulR (Diaz-Uriarte, 2017).

Para la consecución de este objetivo general, nos planteamos los siguientes objetivos parciales:

- Implementación en R (R Core Team, 2018) de la nueva funcionalidad para que el usuario especifique los fitness de los subclones como dependientes de la frecencia genotípica (véase el capítulo Nuevas Funcionalidades y el anexo Material Suplemetario).
- Implementación en C++ 11 de las funciones que realizan los cálculos computacionalmente más costosos debido a su mayor velocidad (véase el capítulo Nuevas Funcionalidades y el anexo Material Suplemetario).
- Manejo del Rcpp (Eddelbuettel and François, 2011; Eddelbuettel, 2013; Eddelbuettel and Balamuta, 2017), paquete de R (R Core Team, 2018) de uso estandarizado en la actualidad que hace de nexo de unión entre este y C++.
- Comprobación de la entera compatibilidad con el código anterior del paquete mediante la superación de la totalidad de los chequeos y su correcta compilación.
- Testeo de la nueva funcionalidad mediante el paquete de R (R Core Team, 2018) testthat (Wickham, 2011) (véase el capítulo Nuevas Funcionalidades y el anexo Material Suplemetario).
- Modificación en Inglés de la documentación del paquete en R documentation format (.Rd) y de las viñetas en R markdown (.Rmd) para documentar la nueva funcionalidad (véase el anexo Material Suplemetario).
- Desarrollo de los casos de estudio (véase el capítulo Casos de estudio) en donde se muestran ejemplos de la nueva funcionalidad disponible.

2

Simuladores de Progresión Tumoral. Estado del Arte

2.1. Introducción

Como argumentamos en el capítulo introductorio de este trabajo (Introducción), los simuladores de la progresión tumoral son herramientas fundamentales en el estudio de la biología del cáncer debido a su poder predictivo y explicativo. Si bien la fiabilidad de las predicciones de las simulaciones deben ser tomadas con cautela, su papel explicativo es incuestionable, siendo de gran ayuda cuando nos enfrentamos a distintos problemas, entre los que destacan la verificación de resultados obtenidos analíticamente, tratar de generar datos para estudiar el comportamiento de métodos estadísticos o examinar modelos sin solución analítica (Diaz-Uriarte, 2017).

En este capítulo se hará una breve panorámica de los distintos simuladores disponibles en la actualidad de funcionalidad similar a OncoSimulR (Diaz-Uriarte, 2017), para a continuación centrarnos en este.

2.2. Simuladores de Progresión Tumoral

En la actualidad existen multitud de simuladores de progresión tumoral disponibles (véanse Peng et al. (2012), Thornton (2014) y la web Genetic Simulations Resources `https://popmodels.cancercontrol.cancer.gov/gsr/`) que permiten el uso de grandes poblaciones y genomas con multitud de genes y que presentan mecanismos flexibles para especificar los efectos de las mutaciones en la tasa de mutación y en el fitness, pudiéndose seguir diferentes esquemas de muestreo, si los efectos de muestreo son relevantes, y mantener la historia filogenética de la evolución clonal, cuando queremos entender la dinámica evolutiva (Diaz-Uriarte, 2017).

Entre los simuladores que presentan las funcionalidades descritas más arriba, comparables con OncoSimulR (Diaz-Uriarte, 2017), podemos destacar simuPOP (Peng et al., 2012), fwdpp (Thornton, 2014), FFPopSim (Zanini and Neher, 2012) y TTP (Reiter et al., 2013), si bien todos ellos carecen de la flexibilidad para especificar los efectos mutacionales, efectos sobre el fitness, efectos del orden de aparición de las mutaciones o diferentes tasas mutacionales para cada gen que presenta OncoSimulR (Diaz-Uriarte, 2017).

simuPOP (Peng et al., 2012) es un simulador de propósito generalista de genética de

poblaciones basado en Python, que emplea un modelo discreto aunque la superposición de generaciones puede simularse haciendo uso del apareamiento no aleatorio. Este software simula poblaciones cuyos individuos presentan su propio genotipo, sexo y otra información adicional de interés, como la edad. La dinámica evolutiva poblacional está determinada por fuerzas genéticas y ambientales, como la mutación, recombinación, migración y cambios en el tamaño poblacional.

Por su parte, fwdpp (Thornton, 2014) es una librería de rutinas de C++, que tiene por intención facilitar el desarrollo de simulaciones bajo el efecto de modelos arbitrarios de fitness y mutaciones. Es un simulador rápido, que consume poca memoria y que provee de cierta flexibilidad para realizar el modelado. Puede emplear dos algoritmos, uno basado en individuos diploides que se reproducen sexualmente, y otro basado en gametos. Thornton (2014) subrayan que este software es muy eficiente con la simulación de grandes poblaciones.

El simulador genérico FFPopSim (Zanini and Neher, 2012) está implementado en C++ con un wrapper de Python2, por lo que se emplea importándolo como módulo de Python. Este simulador presenta ventajas al simular grandes poblaciones con un número moderado de loci debido a que emplea un algoritmo en el que la selección actúa sobre los gametos, reduciéndose el coste computacional al emplear la Transformación de Fourier Rápida.

Por último, TTP (Reiter et al., 2013) es un simulador de la dinámica evolutiva específico para poblaciones tumorales codificado en Java que emplea un modelo de tiempo discreto de procesos de ramificación, en el que los parámetros clave son el fitness landscape, la tasa de mutación y el tiempo medio de división celular. Una de sus mayores fortalezas es que posee una interfaz gráfica amigable, en donde es sencillo realizar las simulaciones e insertar los distintos parámetros.

2.3. OncoSimulR

Entre los simuladores disponibles, destaca el paquete de R (R Core Team, 2018) OncoSimulR (Diaz-Uriarte, 2017), que simula la evolución de poblaciones tumorales con reproducción asexual, modelando el genotipo de los clones como un conjunto de loci bialélicos. Este software permite la definición del fitness, de cada clon como una función arbitraria de las interacciones genéticas entre múltiples genes, o módulos de estos, con una especial atención a la epistasia (relaciones entre genes con un efecto más allá de los puramente aditivos), las restricciones en el orden de aparición de las mutaciones y los distintos efectos que dicho orden de aparición puede implicar.

En lo relativo a las simulaciones, OncoSimulR (Diaz-Uriarte, 2017) emplea dos modelos de progresión tumoral fundamentalmente, un modelo exponencial y una variante del modelo de (McFarland et al., 2013; McFarland, 2014). En ambos modelos tenemos como principales presunciones las siguientes:

- Las células se dividen por reproducción asexual.
- Cada célula presenta un genotipo en el que cada gen sólo puede presentar dos estados, esto es, mutado o no mutado, no contemplándose variaciones alélicas.
- El modelo de tiempo es continuo.
- Sólo se admiten mutaciones de WT a mutado en cada locus, por lo que las mutaciones reversoras no están permitidas.
- Las mutaciones se producen de una en una, nunca varias en el mismo paso.
- La tasa de mutación es despreciable frente a las de natalidad y muerte.

- El fitness de cada subpoblación depende de los efectos que el genotipo y la densidad poblacional ejercen sobre las tasas de nacimiento y muerte (ver más abajo).
- El tamaño poblacional es variable y depende de las tasas de nacimiento y muerte.

En ambos casos, el fitness se especifica a través de la tasa de nacimiento (birth rate b). Si especificamos efectos en los genes que actúan independientemente, entonces b se define mediante el usual modelo multiplicativo (Gillespie, 1993; Beerenwinkel et al., 2007; Zanini and Neher, 2012; Datta et al., 2013) que encontramos en la Ecuación 2.1, en donde s_i es el efecto que sobre el fitness presenta en gen i-ésimo. Este modelo no se satisface si se especifica el fitness landscape o existen restricciones en el orden de mutación o epistasia.

$$b = \prod_{i=1}^{k}(1 + s_i) \tag{2.1}$$

La tasa de muerte (death rate d) es fija y con valor 1 para el modelo exponencial y una expresión dependiente de la densidad poblacional (Ecuación 2.2) en el modelo de McFarland (McFarland et al., 2013; McFarland, 2014), donde N es el taño poblacional y k la población inicial en equilibrio, que se toma por defecto como $k = N_0/(e-1)$, en donde N_0 es la población total inicial. Esto implica que la población inicial está en equilibrio, de modo que $b = d = 1$. Si consideramos la tasa de mutación (μ) despreciablemente pequeña en comparación con las de nacimiento y muerte, hipótesis bastante plausible en poblaciones finitas, de tamaño elevado, en las que la aparición de mutantes es rara, tenemos que la tasa neta de crecimiento para cada clon es $b - d$. Nótese que la tasa neta de crecimiento neta en el caso de emplear el modelo de Mcfarland (McFarland et al., 2013; McFarland, 2014) es una función que depende del tiempo. Esto es debido a que d depende de este como podemos apreciar en Ecuación 2.2 ya que N varía con el tiempo. Es mas, si el fitness depende de la frecuencia genotípica, b también depende indirectamente del tiempo ya que esta varia al cambiar las frecuencias genotípicas relativas con el tiempo en la simulación.

$$d = log(1 + N/k) \tag{2.2}$$

OncoSimulR (Diaz-Uriarte, 2017) emplea en las simulaciones el algoritmo BNB propuesto por Mather et al. (2012), que se basa en un modelo estocástico de tiempo continuo, especialmente pensado para simular la dinámica evolutiva en situaciones con grandes poblaciones que se reproducen asexualmente, con tasas de mutación pequeñas en comparación con las de nacimiento y muerte. La ventaja de este algoritmo sobre otros, como el SSA (Gillespie, 1977), radica en su mayor velocidad en las situaciones descritas más arriba. Esto es debido a que, aunque BNB (Mather et al., 2012) necesita generar varios número aleatorios por iteración en comparación con los 2 de SSA (Gillespie, 1977), este aumento en el costo es pequeño en comparación con los beneficios significativos de saltar sobre las reacciones de nacimiento y muerte en el caso de mutaciones raras, debido a que sólo se actualizan los tamaños subpoblacionales en el momento justo después de darse las mutaciones (Mather et al., 2012).

BNB (Mather et al., 2012) produce una solución exacta cuando la tasas de nacimiento, muerte y mutación permanecen constantes en el proceso evolutivo, obteniéndose una solución aproximada, al menos tan buena como la de otros algoritmos, cuando las tasas anteriores no son constantes, caso en el que es necesario reducir el tiempo entre dos muestreos (Mather et al., 2012). Esto es debido a que si hacemos el tiempo entre dos muestreos suficientemente pequeño, las tasas pueden ser aproximadas como constantes (Mather et al., 2012). Este es el motivo por el cual, cuando empleamos el fitness como dependiente de la frecuencia es necesario reducir el

tiempo entre muestreos y la consecuente actualización del fitness, ya que con ello reducimos la inexactitud del algoritmo.

El núcleo del algoritmo BNB (Mather et al., 2012) se basa en la obtención de una solución exacta para un modelo estocástico de división, muerte y mutación en poblaciones discretas de células. En cada iteración el algoritmo BNB (Mather et al., 2012) usa esta solución exacta para realizar con rapidez los siguientes procesos:

i) Se determina de que clon y en qué momento en el tiempo se generará una nueva célula mutante.

ii) Se actualizan todas las subpoblaciones afectadas hasta el momento anterior a esta mutación.

iii) Se genera una nueva célula mutante que será un nuevo clon o un individuo más si el clon ya existe en la simulación.

iv) Se actualiza el tiempo de la simulación al momento de la mutación.

3

Nuevas Funcionalidades Implementadas y su Testeo

3.1. Introducción

Las novedades implementadas en OncoSimulR (Diaz-Uriarte, 2017) se pueden englobar en tres grupos, las encaminadas a especificar los efectos en el fitness, la evaluación del fitness de uno o todos los genotipos y las simulaciones propiamente dichas (véase Figura 3.1), todas ellas, claro está, en el caso de usar los modelos con el fitness dependiente de la frecuencia.

Como se puede apreciar en la Figura 2.1 el flujo de trabajo necesario para evaluar puntualmente el fitness o realizar simulaciones pasa por la especificación previa del fitness dependiente de la frecuencia. Es decir, si el usuario desea correr una simulación (Simulaciones con Fitness Dependiente de la Frecuencia) o evaluar el fitness en una situación poblacional concreta (Evaluación Puntual del Fitness) es necesario la correcta especificación del fitness (Especificación del Fitness).

Nótese que tanto en la evaluación puntual del fitness, en donde no hay simulación alguna, como en las simulaciones propiamente dichas, el fitness se evalúa, es decir se obtienen los valores de los distintos fitnesses de los correspondientes clones a través de su ecuación. La diferencia estriba en que en la evalución puntual se toman los valores de los tamaños subpoblacionales, aportados por el usuario, para obtener las frecuencias relativas y con ellas, los fitnesses, mientras que en las simulaciones, dichos tamaños subpoblacionales se obtienen como resultado de la propia simulación. En el caso de las simulaciones, la evaluación de los fitnesses se produce cuando fruto de la mutación aparece un nuevo genotipo, y periódicamente, ya que la composición poblacional es variable en el tiempo (véase el capítulo Estado del Arte en Modelos de Progresión Tumoral sección OncoSimulR).

Para instalar la versión implementada en este trabajo del paquete véase el anexo Material Suplementario.

Los parámetros de las funciones empleadas no implementados en este trabajo no están descritas aquí, por lo que para su correcta descripción hay que acudir a la documentación del paquete (Material Suplementario).

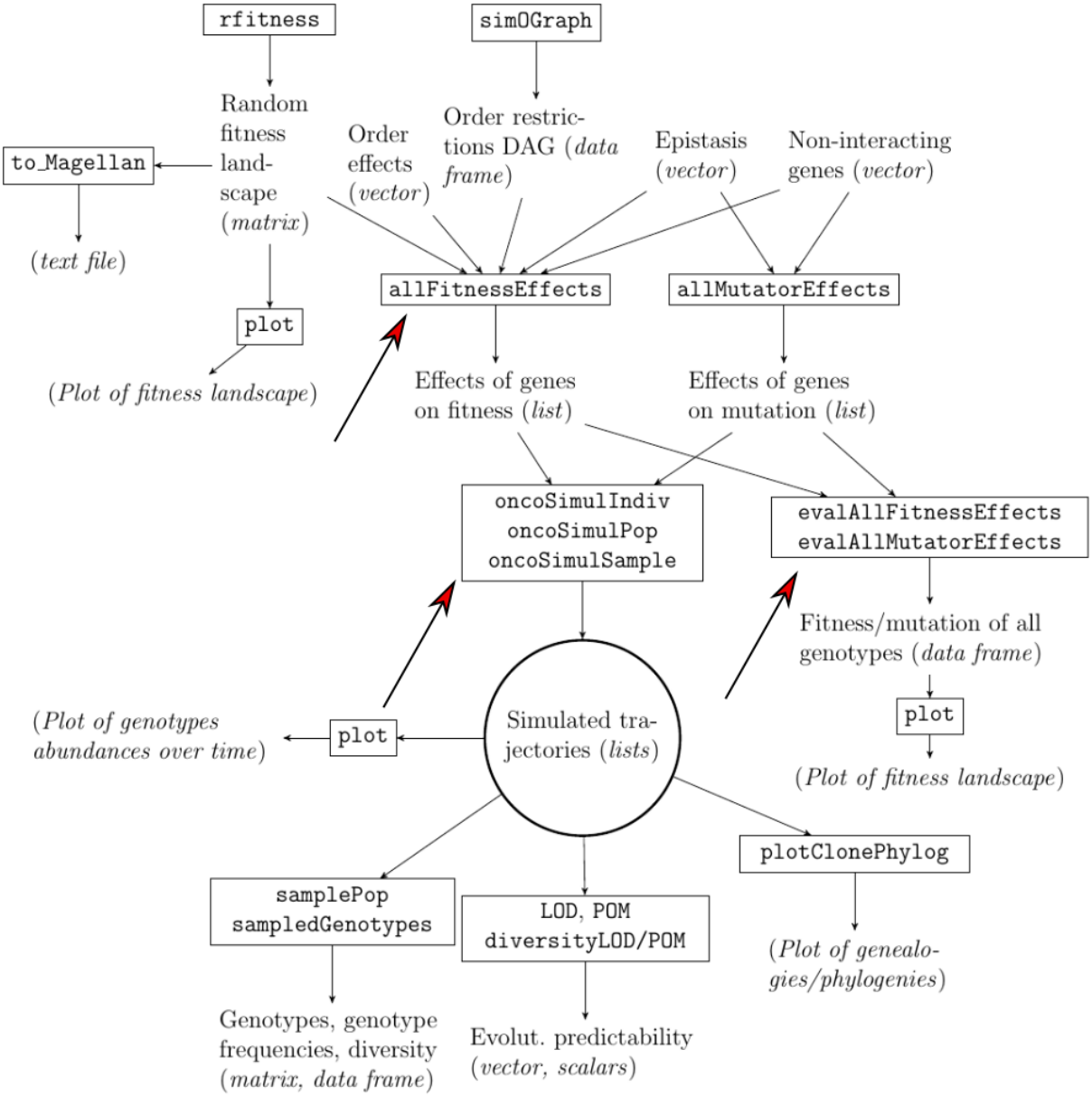


Figura 3.1: Esquema de las principales funciones del paquete OncoSimulR (Datta et al., 2013). Las flechas grandes con punta roja indican los lugares en los que se insertan las nuevas funcionalidades añadidas. Figura tomada de las viñetas del paquete (ver Material Suplemetario).

3.2. Recursos Empleados en la Implementación

Las pruebas del código aislado de R (R Core Team, 2018) se realizaron empleando el IDE RStudio (RStudio Team, 2016) y el de C++ con el IDE Eclipse (Eclipse Fundation, 2018). La inserción del código escrito en ambos lenguajes en el código previo se realizó mediante el editor de texto Atom (GitHub, 2018a). Las series de compilaciones y testeos de la totalidad del paquete se ejecutaron en el terminal. Las pruebas de la totalidad del código también se realizaron con RStudio (RStudio Team, 2016). El control de versiones se realizó empleando GitHub (GitHub, 2018b) en el terminal. La edición de la documentación del paquete y las viñetas se realizó con RStudio (RStudio Team, 2016).

Una de las cuestiones clave en la implementación de la nueva funcionalidad del fitness dependiente de la frecuencia genotípica, fue el empleo de la librería de parseo y evaluación de

expresiones matemáticas de C++ ExprTk (Partow, 2018). Ella permite que el usuario pueda escribir en R (R Core Team, 2018) las ecuaciones del fitness, como una cadena de caracteres, que será posteriormente parseada y evaluada con el código en C++ cuando se pretenda hacer una evaluación puntual del fitness (Evaluación Puntual del Fitness) o bien cuando se corre una simulación (Simulaciones con Fitness Dependiente de la Frecuencia), como ya hemos comentado en la sección introductoria a este capítulo (Introducción).

ExprTk (Partow, 2018) tiene una licencia MIT Open Source (`https://opensource.org/licenses/MIT`) por lo que puede ser empleada sin restricciones en cualquier proyecto de desarrollo de software. En este caso, gracias a su flexibilidad de empleo implementamos su uso como header (exprtk.hpp) en el código en C++. Para ello, siguiendo las recomendaciones de Eddelbuettel (2013) se situó el header en el directorio OncoSimul/OncoSimulR/inst/miscell, especificándose su localización el los fichero Makevars y Makevars.win localizados en OncoSimul/OncoSimulR/src. Este proceder aumenta el tamaño del paquete, pero no requiere de la descarga de la librería por parte del usuario.

En la implementación del fitness dependiente de la frecuencia en el paquete OncoSimulR (Diaz-Uriarte, 2017) se adaptó el uso de ExprTk (Partow, 2018) de modo que esta es empleada cuando el fitness se evalúa. Las líneas generales del flujo de trabajo son las siguientes:

i) Se especifican las ecuaciones como cadena de caracteres en R (R Core Team, 2018) (Especificación del Fitness).

ii) Mediante Rcpp (Eddelbuettel and François, 2011; Eddelbuettel, 2013; Eddelbuettel and Balamuta, 2017) se pasan estas ecuaciones a C++, en donde se almacenan como cadenas de caracteres en un objeto que contiene las especificaciones del fitness landscape y las variables, que son las frecuencias relativas de los clones (Especificación del Fitness).

iii) Para evaluar cada uno de los fitnesses se construye una tabla con las variables y su valor numérico, que se obtiene de interrogar los genotipos presentes en cada momento de evaluación.

iv) Con los valores de la tabla anterior se parsea y evalúa la expresión contenida en la ecuación que especifica el fitness.

v) Se repiten los pasos iii) y iv) tantas veces como clones queramos evaluar.

vi) Se devuelven a R (R Core Team, 2018) los resultados mediante Rcpp (Eddelbuettel and François, 2011; Eddelbuettel, 2013; Eddelbuettel and Balamuta, 2017).

Los símbolos y operaciones matemáticas permitidas en las ecuaciones, gracias a los cuales esta herramienta es muy completa y flexible, son las disponibles en la librería ExprTk (Partow, 2018) (Véase Material Suplementario en donde se halla un enlace a la documentación de la librería).

3.3. Especificación del Fitness

La especificación de los efectos sobre el fitness se realiza en el paquete con la función *allFitnessEffects*. En el caso de especificar el fitness dependiente de la frecuencia, esta función se emplea como podemos apreciar en la Code Box 3.1.

El mapeo genotipo fitness se especifica mediante un data frame (*genoFitness*) y en la función *allFitnessEffects* basta con señalar *frequencyDependentFitness* = TRUE (Ver Material Suplementario para ir a la documentación del paquete y viñetas).

Podemos observar en el código (Code Box 3.1) que la columna del fitness es un vector de caracteres en el que la entrada que se corresponde con la fila de cada genotipo representa la ecuación que especifica el fitness como función de las frecuencias genotípicas. Estas se representan por f_*, en donde * va a especificar el genotipo, de modo que f_ será la frecuencia relativa del WT, f_A o f_1 la del genotipo que presenta únicamente el gen A mutado, f_A_B o f_1_2, será la del genotipo con los genes A y B mutados, etc.

```
## Genotype-Fitness mapping
r <- data.frame(Genotype = c("WT", "A", "B", "A, B"),
                Fitness = c("2 + f_",
                    "3/(1 + f_1^2)",
                    "3/(1 + f_2^2) ",
                    "4/(1 + f_1_2^2)"),
                stringsAsFactors = FALSE)
## Fitness Efects specification
afe <- allFitnessEffects(genotFitness = r,
                         frequencyDependentFitness = TRUE,
                         spPopSizes = c(5000, 2000, 3000, 500))
```

Code Box 3.1: Ejemplo de especificación del fitness en el caso de que este dependa de la frecuencia genotípica.

El parámetro *spPopSizes* contiene las frecuencias absolutas de los distintos clones presentes en la población. Esta especificación es innecesaria si posteriormente se pretende realizar una simulación, no siendo el caso si queremos realizar una evaluación puntual del fitness (ver secciones siguientes Evaluación Puntual del Fitness y Simulaciones con Fitness Dependiente de la Frecuencia).

3.4. Evaluación Puntual del Fitness

Evaluar el fitness de manera puntual de uno o más genotipos cuando estos dependen de las frecuencias genotípicas requiere el conocimiento de dichas frecuencias, y por tanto de los tamaños subpoblacionales. Por ello, se modificó la función *allFitnessEffects* para que pudiera contener un vector de números (*spPopSizes*), facilitado por el usuario, que contuviera los tamaños o frecuencias absolutas de las subpoblaciones clonales, como se puede apreciar en la Code Box 3.1. Para realizar la evaluación puntual del fitness, el paquete cuenta con la función *evalGenotype*, que evalúa el genotipo que se especifique, y *evalAllGenotypes*, que los evalúa todos. Ambas funciones fueron modificadas para aceptar un objeto *fitnessEffects* que contuviera las especificaciones necesarias para realizar su función en el caso que nos ocupa en este trabajo. Así, la interfaz del programa fue adaptada para contener las nuevas funcionalidades sin la necesidad de producir otras funciones. Un ejemplo de esto se puede apreciar en la Code Box 3.2.

```
## Genotype "A, B" evaluation
evalGenotype(genotype = "A, B", fitnessEffects = afe)
## All genotypes evaluation
evalAllGenotypes(fitnessEffects = afe)
## Plot fitness landscape
plotFitnessLandscape(evalAllGenotypes(fitnessEffects = afe))
```

Code Box 3.2: Evaluación de Genotipos y plot del fitness landscape (Figura 2.2).

En la Figura 3.2 podemos encontrar una foto fija del fittness landscape que se obtiene con las subpoblaciones definidas en el Code Box 3.1. En el caso del fitness dependiente de la frecuencia genotípica, el fitness landscape varía con los tamaños subpoblacionales, y por tanto también con

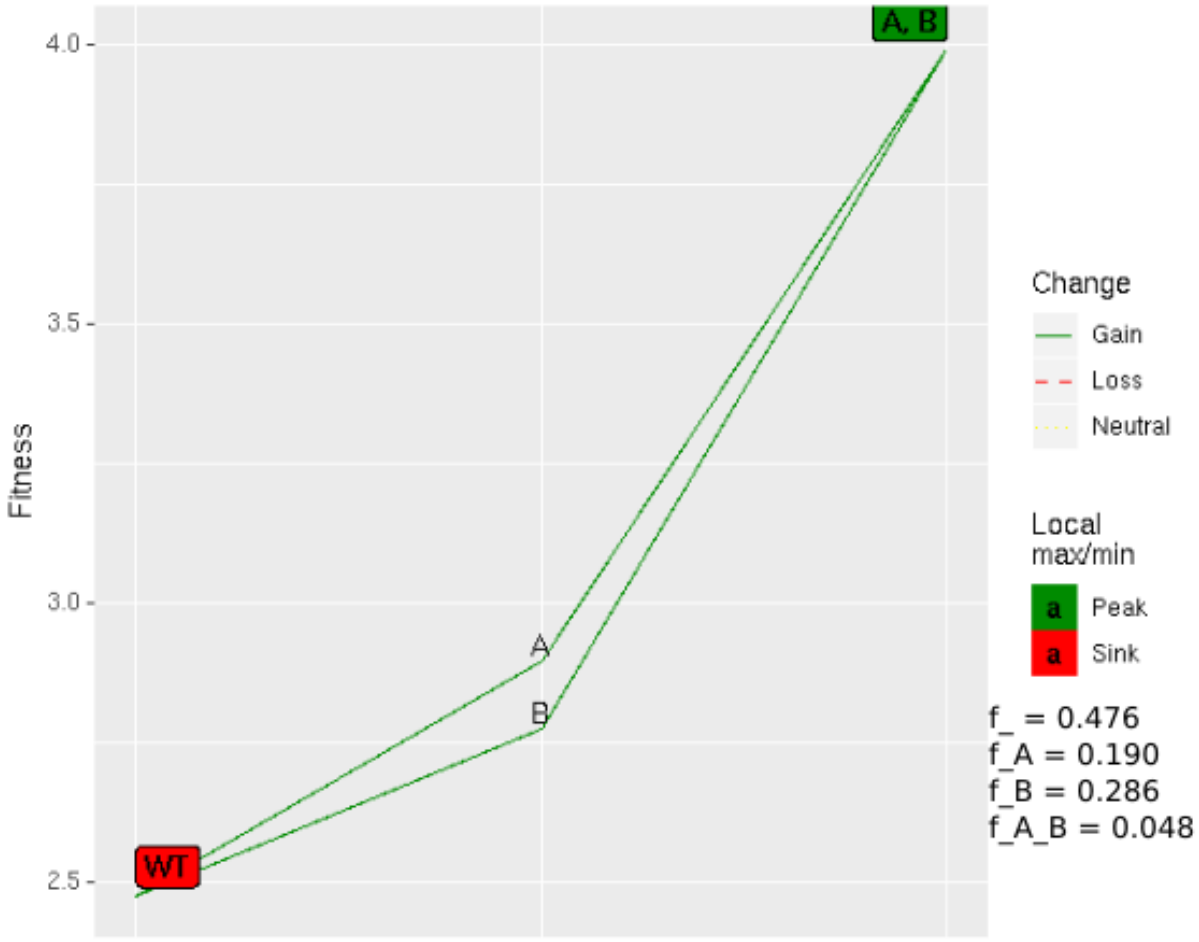


Figura 3.2: Fitness landscape plot definido en la Code Box 3.2.

sus frecuencias relativas, por lo que lo representado aquí sería como una instantánea tomada en el momento en el que los tamaños de los diferentes clones son los especificados.

En el empleo de estas dos funciones en R (R Core Team, 2018), se pasan a C++ las especificaciones del fitness y este código se encarga de la evaluación, empleando Rcpp (Eddelbuettel and François, 2011; Eddelbuettel, 2013; Eddelbuettel and Balamuta, 2017) como enlace entre ambos lenguajes y ExprTk (Partow, 2018) para parsear y evaluar las ecuaciones. El resultado se exporta de nuevo a R (R Core Team, 2018), mediante Rcpp (Eddelbuettel and François, 2011; Eddelbuettel, 2013; Eddelbuettel and Balamuta, 2017).

Para más información y ejemplos de uso, véase Material Suplementario para ir a la documentación del paquete y las viñetas.

3.5. Simulaciones con Fitness Dependiente de la Frecuencia

Como podemos observar en la Figura 3.1, en las simulaciones, OncoSimulR (Diaz-Uriarte, 2017) presenta tres funciones:

- *oncoSimulIndiv*: esta función simula una única trayectoria evolutiva.

- *oncoSimulPop*: simula tantas trayectorias evolutivas como le especifiquemos, con las mismas condiciones, es decir, llama a la anterior función el número de veces especificado.

- *oncoSimulSample*: esta función realiza el mismo trabajo que la anterior, pero el resultado es ligeramente diferente, devolviendo solo el resumen final y el resultado de un muestreo de la población.

Siguiendo con el ejemplo de las anteriores Code Boxes (Code Box 3.1 y Code Box 3.2), para simular mediante el modelo de McFarland una única trayectoria evolutiva, lo podemos hacer mediante el código de Code Box 3.3.

En este caso utilizamos la interfaz ya existente en OncoSimulR (Diaz-Uriarte, 2017) para realizar las simulaciones, ampliandola para contener el objeto *afe* que incorpora en este caso todas las especificaciones para el caso de fitness dependiente de la frecuencia genotípica.

```
## For reproductibility
set.seed(1)
## Run one trayectory simulation
osi <- oncoSimulIndiv(afe,
                      model = "McFL",
                      onlyCancer = FALSE,
                      finalTime = 5000,
                      verbosity = 0,
                      mu = 1e-6,
                      initSize = 5000,
                      keepPhylog = FALSE,
                      seed = NULL,
                      detectionProb = NA,
                      detectionSize = NA,
                      errorHitMaxTries = FALSE,
                      errorHitWallTime = FALSE)
## See results
osi
## Plot genotypes sizes along time
plot(osi, show = "genotypes", type = "line", xlim = c(-1, 2200))
```

Code Box 3.3: Evaluación de Genotipos y plot del fitness landscape (Figura 3.3).

En la Figura 3.3 se puede observar la evolución de los tamaños subclonales a lo largo del tiempo de la simulación. En ella se puede apreciar como en la situación especificada al cabo de cierto tiempo desaparecen los genotipos WT y B, manteniéndose B y AB con un tamaño mas o menos constante, si bien realizan pequeñas fluctuaciones.

Para más información sobre las funciones y sus parámetros y ejemplos de uso véase Material Suplementario para ir a la documentación del paquete y a las viñetas.

3.6. Nuevos Ficheros de Testeo

Testar los programas es una cuestión fundamental en el desarrollo de software de calidad. Por esto, el paquete OncoSimulR (Diaz-Uriarte, 2017) cuenta con más de 3000 test que se ejecutan mediante el paquete de R (R Core Team, 2018) thestthat (Wickham, 2011). Para mantener este estándar de calidad, el desarrollo de las nuevas funcionalidades vino acompañado de la programación de tres nuevos ficheros de testeo, uno para cada grupo de las nuevas funcionalidades, con los que se cubre la totalidad del nuevo código implementado:

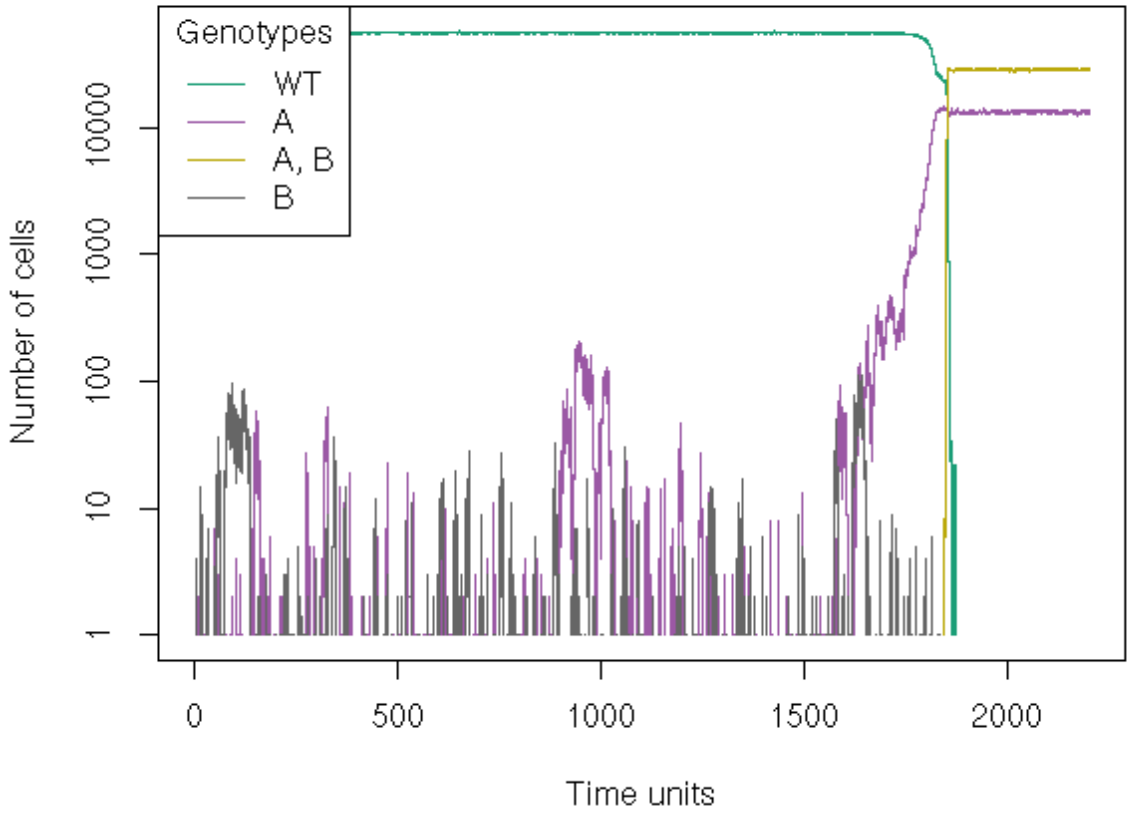


Figura 3.3: Tamaño de las subpoblaciones clonales a lo largo del tiempo (plot definido en la Code Box 3.3).En este caso encontramos un fenómeno curioso en el que los clones A y AB conviven en el tumor.

1. *test.allFitnessEffectsFDF.R*: este fichero testea la especificación del fitness cuando el fitness es dependiente de la frecuencia. En concreto se verifica el correcto funcionamiento de los siguientes aspectos:

   a) El objeto *genofitness* es un data.frame, que nunca puede ser nulo.

   b) La última columna de *genofitness*, que contiene las ecuaciones del fitness, es un vector de caracteres.

   c) La última columna de *genofitness* contiene las variables (frecuencias gnotípicas relativas) correctamente especificadas.

   d) Si *genofitness* es un data.frame de dos columnas, entonces la primera debe contener una correcta especificación de los genotipos con letras o nombres para los distintos genes.

   e) Si *genofitness* es un data.frame de más de dos columnas, las n-1 primeras deben de contener una correcta especificación de los genotipos, con filas de 0 o 1 según en esa posición el gen esté mutado o no.

   f) Comprobación de que la especificación del fitness por el método tratado en 1.e y 1.d son equivalentes.

   g) Todos los elementos del objeto *allFitnessEffects* que produce la función homónima tienen la clase correcta.

h) El objeto *spPopSizes* no es necesario, y se obtiene un warnnig, cuando no estamos en una situación en la que el fitness dependa de la frecuencia.

i) Cuando la última columna de *genofitness* es un factor, este convierte automáticamente a un vector de caracteres. Este es un problema corriente cuando se construyen data.frames y no se explicita que los strings no son factores.

j) Los resultados son idénticos cuando utilizamos las variables de las ecuaciones de fitness con letras (f_, f_A, f_A_B, ...) o con números (f_, f_1, f_1_2, ...).

2. *test.evaluatingGenotypesFDF.R*: para testar la evaluación puntual de los genotipos. En este fichero se comprobaron las siguientes cuestiones:

   a) Se comprobó que cuando se evalúa puntualmente un solo genotipo el resultado es idéntico cuando se especifica el genotipo mediante vectores numéricos convencionales o con un string que contiene los genes mutados como letras.

   b) Cuando se evalúa el WT su genotipo se puede especificar como 0 o Root y el resultado es idéntico.

   c) Al evaluar un genotipo con más de una posición mutada, si especificamos este mediante un vector convencional, el 0 no puede estar incluido.

   d) En el caso de no estar en una situación de fitness dependiente de la frecuencia genotípica y tratar de evaluar el fitness del genotipo WT poniéndolo en el parámetro *genotype* se produce un error, consistentemente con el funcionamiento del paquete antes de las modificaciones de este trabajo.

   e) Si no pasamos el objeto *spPopSizes* y tratamos de evaluar el fitness cuando este es dependiente de la frecuencia se obtiene un error.

   f) Cuando tratamos de evaluar un genotipo incorrectamente especificado o con genes no contenidos en el objeto *fitnessEffects* se obtiene un error.

   g) Se comprueba que el resultado de evaluar todos los genotipos uno a uno mediante *evalGenotype* es idéntico a evaluarlos todos a la vez con *evalAllGenotypes* en el caso de que el fitness dependa de la frecuencia.

3. *test.Z-oncoSimulIndivFDF.R*: para el caso de la simulaciones. En este fichero se empleó la semilla (seed) para testar los valores de algunos de los resultados al eliminar los efectos aleatorios fijando el punto de partida del algoritmo generador de números pseudoaleatórios. En este caso se testó que funcionaban perfectamente según lo deseado las cuestiones que a continuación se enumeran:

   a) Las clases de todos los objetos generados con la salida de *oncoSimulIndiv* se corresponden con los esperados.

   b) En la ejecución se obtienes los valores esperados para una trayectoria evolutiva simulada en unas condiciones concretas.

   c) Se comprueba que si empleamos el modelo de Bozic (Bozic et al., 2010) se obtienen errores durante la simulación que hacen que esta aborte, debido al problema de que la tasa de muerte se hace cero en la simulación. Para más información sobre este tema, véanse las viñetas (Material Suplementario).

Con el testeo se cubren dos objetivos fundamentales, la comprobación de que las novedades funcionan según lo esperado, por un lado, y por otro, se facilita la compatibilidad e integración de posteriores añadidos al código. Los beneficios del primer objetivo son obvios, ya que se verifica el correcto funcionamiento del código implementado. Con el segundo objetivo, se permite una correcta comprobación de que si añadiéramos nuevo código, este debiera integrarse con el ya

existente y por tanto no fallar en los tests. Si no fallan los test, podemos tener bastante evidencia de una correcta integración. Si esto no ocurriera, comprobando donde falla podemos acotar el terreno para el debugging, lo cual es siempre muy positivo.

Para acceder al repositorio del paquete y ver estos ficheros véase Material Suplementario.

4

Casos de Estudio

4.1. Introducción

En este capítulo trataremos en detalle dos casos de estudio, para poner de manifiesto, mediante ejemplos, el alcance de la nueva funcionalidad implementada en el paquete OncoSimulR (Diaz-Uriarte, 2017).

En ambos casos emplearemos el modelo de McFarland (McFarland et al., 2013), ya que lo consideramos el más realista, debido a que contempla la tasa de mortalidad como dependiente de la densidad, y en el caso del fitness dependiente de la frecuencia genotípica, que es lo que este TFM aporta como novedad, también una tasa de natalidad dependiente de dicha frecuencia genotípica. Por tanto, emplearemos este modelo para correr nuestras simulaciones como dependientes de la frecuencia genotípica a la par que de la densidad poblacional.

4.2. Caso de Estudio I

Axelrod et al. (2006) pusieron de manifiesto que los distintos clones presentes en un tumor podían colaborar entre sí mediante el aporte al medio extracelular de productos difusibles, del tipo de los factores de crecimiento. De este modo, los distintos subclones pueden cooperar entre ellos, hasta que la totalidad del tumor adquiera las marcas del cáncer (Hanahan and Weinberg, 2000, 2011) y el estado de malignidad pueda ser entendido como una propiedad emergente del sistema.

Para este ejemplo, supodremos que estamos interesados en dos genes (A y B) y por tanto existen 4 genotipos posibles (WT, A, B y AB). La tasa de natalidad del genotipo WT es constante y vale 1. De este modo la población parte del equilibrio, como se justificó en la sección OncoSimulR del capítulo Simuladores de Progresión Tumoral. Los mutantes simples, presentan una tasa de natalidad más elevada que el genotipo WT, dependiendo de la frecuencia de un modo lineal $(1 + 3(f\_A + f\_B + 2f\_A\_B))$, pero no solo de su propia frecuencia, si no también de la de todos los mutantes de un modo aditivo. El doble mutante presenta una tasa de natalidad que se comporta de la misma forma que los mutantes simples, pero su fitness es el doble de los simples $(2 + 6(f\_A + f\_B + 2f\_A\_B))$.

La cooperación entre los distintos subclones mutantes viene dada por el componente ($f\_A + f\_B + 2f\_A\_B$). Esta expresión puede ser entendida del siguiente modo. Supongamos que el mutante A produce gran cantidad de un factor de crecimiento A necesario para el desarrollo tumoral, pero escasa cantidad de otro B, también necesario, que sin embargo es producido en exceso por el mutante B, que a su vez produce poco factor A. El doble mutante, por otro lado, produce en exceso de los dos factores de crecimiento, de ahí que su frecuencia esté multiplicada por dos, mientras que el WT, produce cantidades basales despreciables de ambos frente a las grandes concentraciones producidas por los mutantes. Así, la presencia de los subclones mutantes favorece el fitness de los otros, estableciéndose una relación mutualista. La diferencia de fitness entre los cuatro genotipos puede ser entendida como una diferente susceptibilidad a los factores de crecimiento. Si suponemos, a modo de ejemplo, que la producción de estos es directamente proporcional a la cantidad de receptores presentes en la membrana celular, podemos explicar que el fitness del WT sea menor que el de los mutantes simples y menor que el del doble. El doble mutante, produce el doble de factores de crecimiento y es doblemente susceptible, por lo su tasa de natalidad también se duplica.

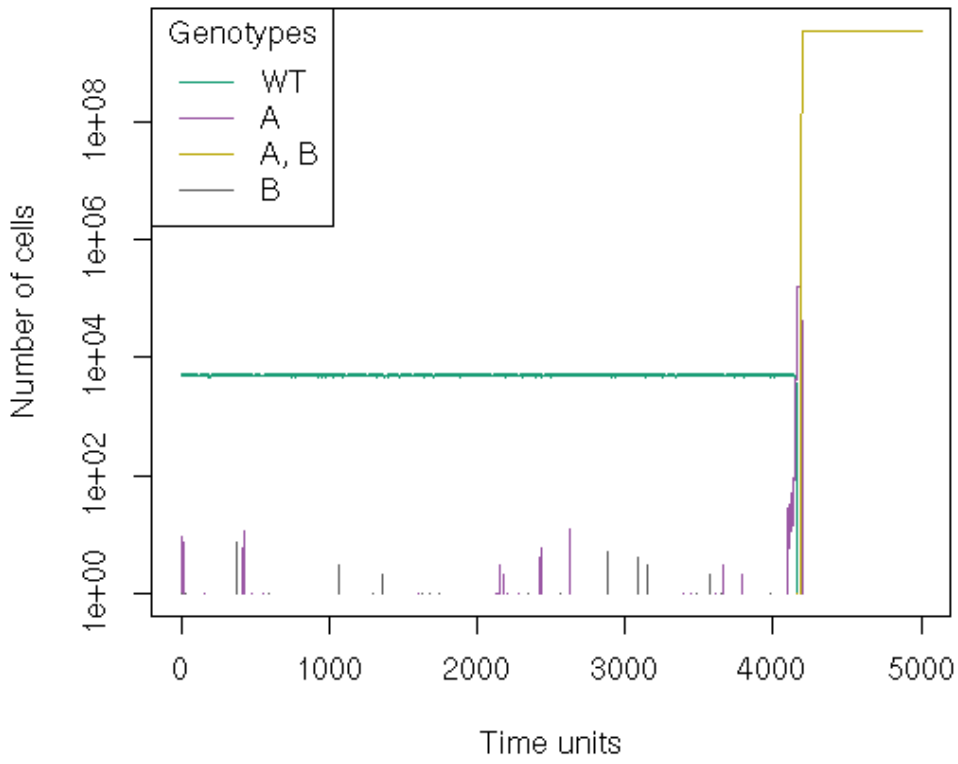
```
r <- data.frame(Genotype = c("WT", "A", "B", "A, B"),
                Fitness = c("1",
                            "1 + 3*(f_A + f_B + 2*f_A_B)",
                            "1 + 3*(f_A + f_B + 2*f_A_B)",
                            "2 + 6*(f_A + f_B + 2*f_A_B)"),
                stringsAsFactors = FALSE)

afe <- allFitnessEffects(genotFitness = r,
                         frequencyDependentFitness = TRUE)
set.seed(1)
osi <- oncoSimulIndiv(afe,
                      model = "McFL",
                      onlyCancer = FALSE,
                      finalTime = 5000,
                      verbosity = 0,
                      mu = 1e-6,
                      initSize = 5000,
                      keepPhylog = TRUE,
                      seed = NULL,
                      detectionProb = NA,
                      detectionSize = NA,
                      errorHitMaxTries = FALSE,
                      errorHitWallTime = FALSE)
osi
plot(osi, show = "genotypes", type = "line")
plotClonePhylog(osi, N = 0)
```

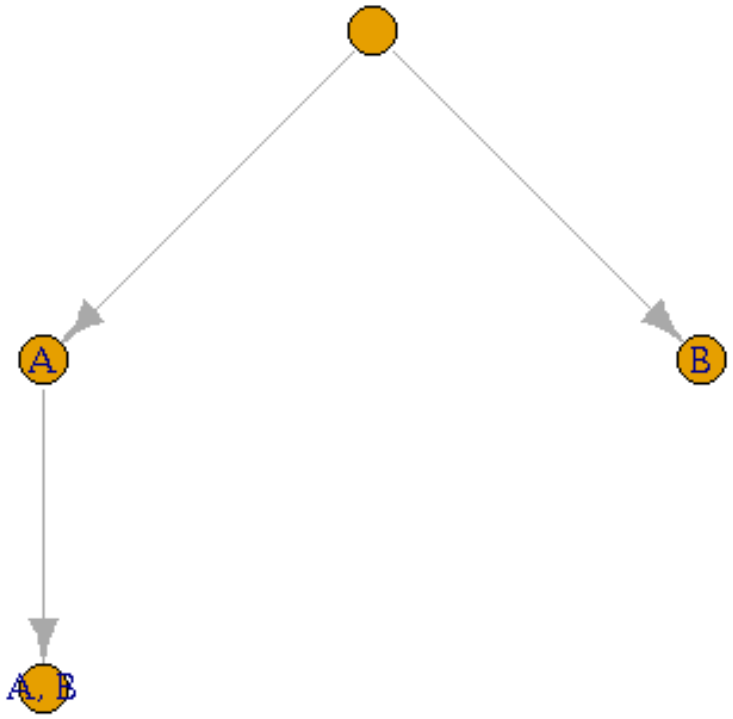
Code Box 4.1: Código para simular el caso de estudio I

El código necesario para correr esta simulación lo podemos encontrar en la Code Box 4.1. El gráfico del tamaño de los distintos genotipos a lo largo del tiempo lo podemos apreciar en la Figura 4.1.a y en Figura 4.1.b el gráfico de la filogenia de los distintos subclones. Este último gráfico se presenta para dar a conocer la posibilidad de su realización de manera sencilla, si bien no es muy relevante en este ejemplo. La composición de la población final se puede encontrar en la Tabla 3.1.

La Figura 4.1.a muestra que cuando la frecuencia de uno de los mutantes simples, en este caso el A, supera la del WT, este desaparece debido a que la tasa de muerte supera a la de nacimiento por efecto del aumento de la densidad poblacional. Durante un tiempo conviven el mutante simple A y el doble mutante, pero a a partir de que este último, de mayor fitness, supera



(a)



(b)

Figura 4.1: a) Tamaño de los genotipos frente al tiempo. b) Filogenia de los distintos subclones.

Genotype	N
WT	0
A	0
B	0
A, B	3499772701

Tabla 4.1: Composición final de la población en el Caso de Estudio I.

en número al simple, la tasa de muerte aumenta por efecto del gran aumento de AB, superando la de nacimiento de los mutantes simples, y este se extingue. Por su parte, el doble mutante crece hasta que la tasa de nacimiento y la de muerte se igualan y su tamaño permanece constante, comportándose como un modelo logístico. Por tanto, en la situación simulada, desaparecen todos los subclones a excepción del doble mutante, que presenta un mayor fitness y una vez que aparece desplaza a todos los demás, llevándolos a la extinción, como puede apreciarse en la Tabla 4.1.

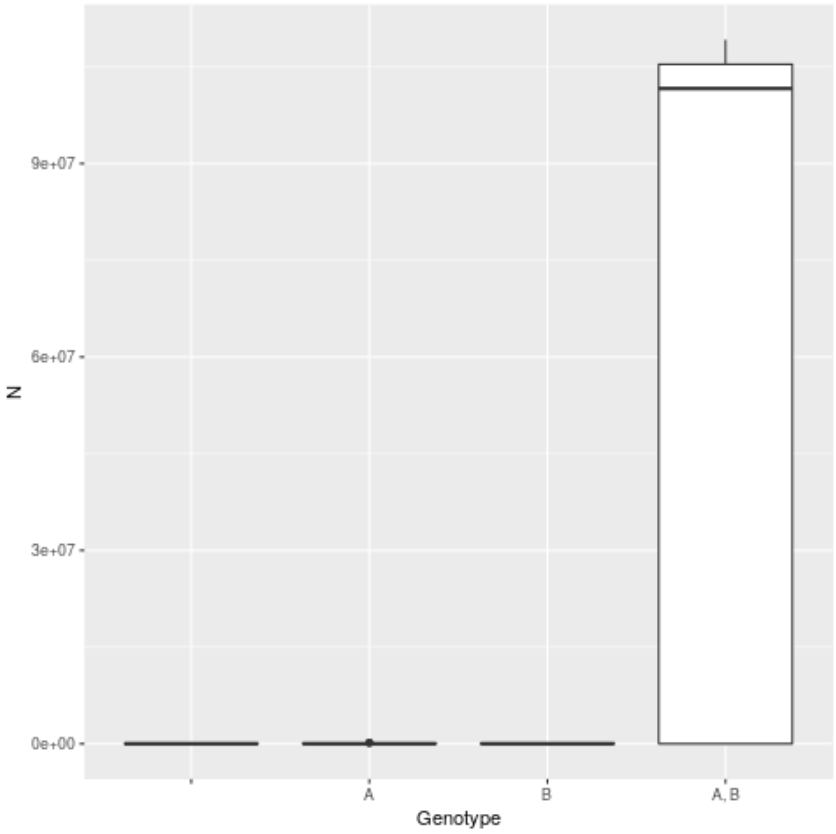


Figura 4.2: Boxplot con la composición poblacional tras correr el Caso de Estudio I 1000 veces. En este caso se corrieron las simulaciones utilizando los parámetros de parada por defecto, para disminuir el tiempo de computación, lo cual interrumpe la simulación con anterioridad cuando la población alcanza un determinado tamaño.

La filogenia de los distintos genotipos (Figura 4.1.b) nos muestra que el doble mutante, en esta simulación, aparece mutando desde el mutante A. Esto, podría inducir a pensar que actuando contra el mutante A, podríamos controlar la progresión del tumor al estado de mayor malignidad AB. Sin embargo, si corremos de nuevo esta simulación, obtenemos diferentes resultados, en los que se observa (resultados no mostrados) que sólo es necesario, como es lógico, la presencia de uno de los mutantes simples para que se pueda dar el doble. Tanto el mutante A como el B puede aparecer desde WT con igual probabilidad, ya que la tasa de mutación es idéntica para ambos ($mu = 10^{-6}$). Esto nos lleva a plantearnos la estabilidad de los resultados y qué

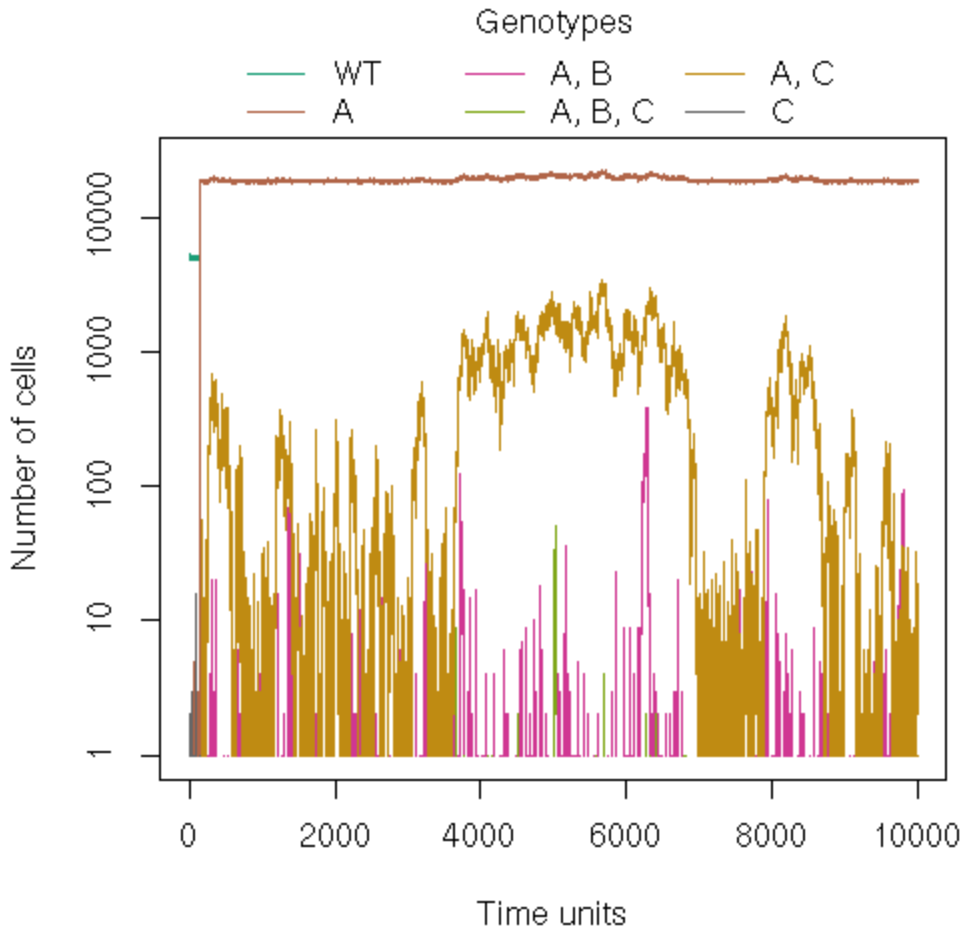
ocurriría si corriéramos un número elevado de veces las simulaciones. Para responder a esta pregunta corrimos la simulación 1000 veces y los resultados pueden apreciarse en la Figura 4.2, que nos muestra que en cualquier caso la población evoluciona al estado de mayor malignidad con el doble mutante dominando enteramente la población en las condiciones simuladas. En la misma figura también podemos apreciar que los efectos estocásticos producen una población final dominada por el mutante AB, pero de tamaño bastante variable.

4.3. Caso de Estudio II

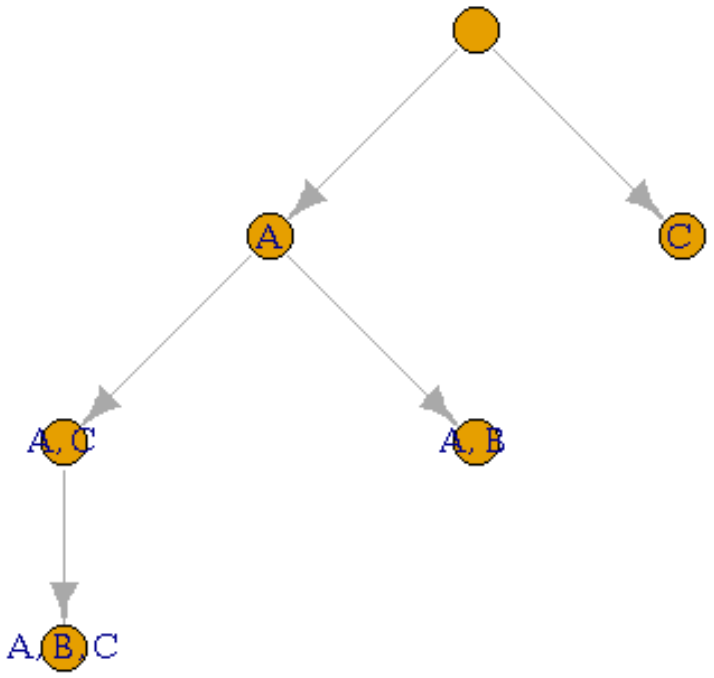
En el Caso de Estudio II trataremos de simular una escenario algo más complejo. En este caso, tenemos 3 genes y por tanto 8 genotipos posibles. El WT presenta la misma tasa de natalidad que en el caso anterior para que la simulación parta del equilibrio. Aquí vamos a suponer que los clones con la posición C mutada producen un factor difusible proliferativo y que los clones con B mutado producen otro antiproliferativo, pero cuya acción es la mitad de efectiva que el proliferativo. Los clones con A mutado, por su parte, presentan el doble de susceptibilidad a ambos factores. En este caso, consideraremos la tasa de mutación del gen C 10 veces superior a la de los otros 2 genes. La codificación de esta situación la podemos observar en Code Box 4.2, en donde se aprecia que los clones que no presentan A mutado tienen por tasa de natalidad f y los que si la presentan f2, que no es otra cosa que el doble del valor f. El valor f incluye el aporte al fitness de la presencia de los clones que producen el factor proliferativo, con signo positivo, y el aporte de los que producen el factor antiproliferativo, con signo negativo.

```
f <- "1 + f_C + f_A_C + f_B_C + f_A_B_C - 0.5*(f_B + f_A_B + f_B_C + f_A_B_C)"
f2 <- "2*(1 + f_C + f_A_C + f_B_C + f_A_B_C - 0.5*(f_B + f_A_B + f_B_C + f_A_B_C))"
r <- data.frame(Genotype = c("WT",
                             "A", "B", "C",
                             "A, B", "A, C", "B, C",
                             "A, B, C"),
                Fitness = c("1",
                            f2, f, f,
                            f2, f2, f,
                            f2),
                stringsAsFactors = FALSE)
afe <- allFitnessEffects(genotFitness = r,
                         frequencyDependentFitness = TRUE)
set.seed(1)
osi <- oncoSimulIndiv(afe,
                      model = "McFL",
                      onlyCancer = FALSE,
                      finalTime = 10000,
                      verbosity = 0,
                      mu = c(A = 1e-6, B = 1e-6, C = 1e-5),
                      initSize = 5000,
                      keepPhylog = TRUE,
                      seed = NULL,
                      detectionProb = NA,
                      detectionSize = NA,
                      errorHitMaxTries = FALSE,
                      errorHitWallTime = FALSE)
osi
plot(osi, show = "genotypes", type = "line", xlim = c(-1, 170))
plotClonePhylog(osi, N = 0)
```

Code Box 4.2: Código para simular el caso de estudio II



(a)



(b)

Figura 4.3: a) Tamaño de los genotipos frente al tiempo. b) Filogenia de los distintos subclones.

El código necesario para simular esta situación se puede encontrar la Code Box 4.2. El gráfico del tamaño de los distintos genotipos a lo largo del tiempo puede ver en la Figura 4.2.a) y en Figura 4.2.b) el gráfico de la filogenia de los distintos subclones. La composición de la población final está en la Tabla 4.2. Al igual que en el caso anterior, la simulación se corrió 1000 veces para ver si el resultado es estable y el boxplot se puede ver en la Figura 4.4.

Genotype	**N**
WT	0
A	18573
B	0
C	0
A, B	0
A, C	0
B, C	0
A, B, C	0

Tabla 4.2: Composición final de la población en el Caso de Estudio II.

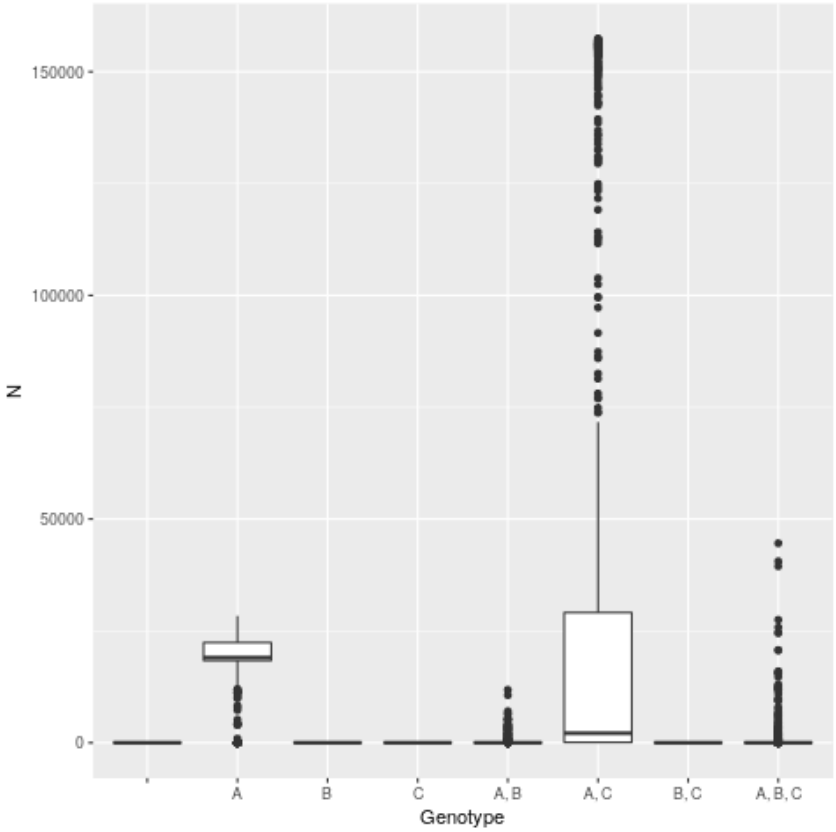
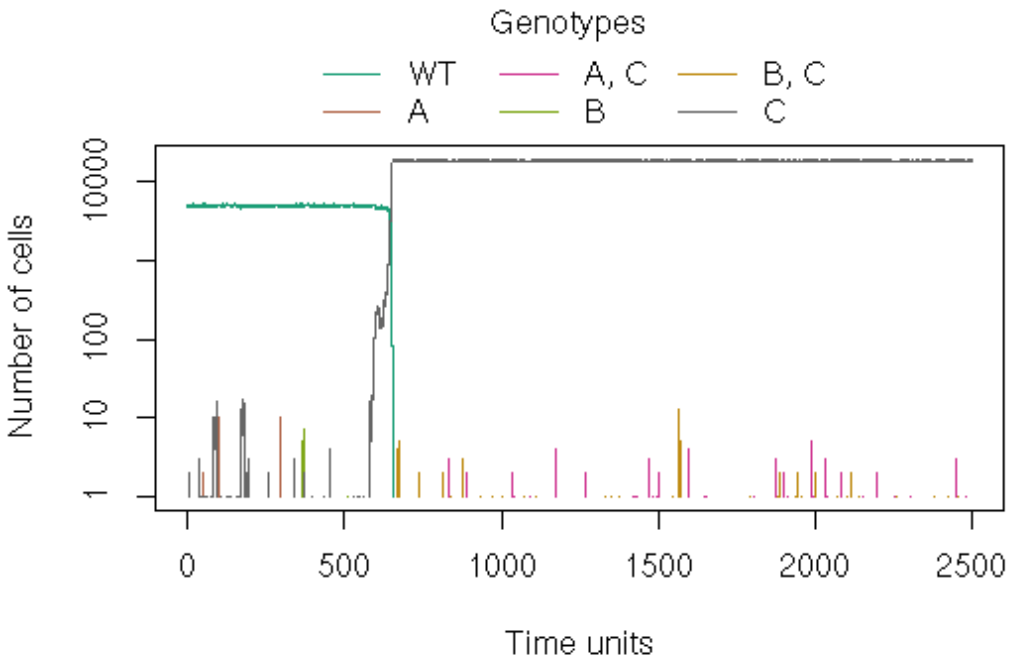
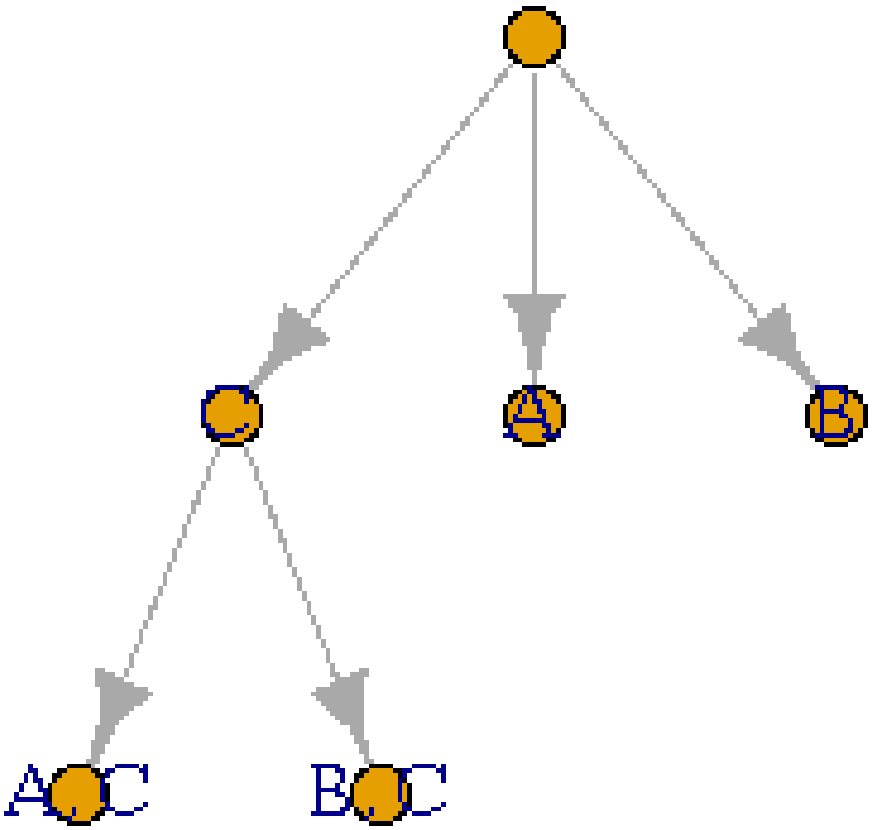


Figura 4.4: Boxplot con la composición poblacional tras correr el caso de Estudio II 1000 veces. En este caso se corrieron las simulaciones utilizando los parámetros de parada por defecto, para disminuir el tiempo de computación, lo cual interrumpe la simulación con anterioridad cuando la población alcanza un determinado tamaño.

Los resultados indican que con la situación simulada, llegamos a una población final en la que el mutante A domina enteramente (Figura 4.3.a y Tabla 4.2), apareciendo los mutantes AB y AC y más raramente ABC, si bien estos no llegan a dominar la situación y desaparecen al verse desplazados por efecto de la alta densidad poblacional consecuencia del elevado número del clon A. A pesar de que el gen C presenta ua tasa de mutación 10 veces mayor que los demás, el mayor fitness de los clones con A mutado parece inclinar la balanza en favor de la aparición



(a)



(b)

Figura 4.5: a) Tamaño de los genotipos frente al tiempo. b) Filogenia de los distintos subclones.

Genotype	N
WT	0
A	0
B	0
C	18642
A, B	0
A, C	0
B, C	0
A, B, C	0

Tabla 4.3: Composición final de la población en el Caso de Estudio II con fármaco anti mutantes C.

de estos. Así, los dobles y triple mutante en la situación concreta simulada aparecieron siempre desde A, como podemos apreciar el la filogenia mostrada en Figura 4.3.b.

Simulando esta situación 1000 veces para estudiar los efectos estocásticos sobre la población final obtenemos el gráfico de la Figura 4.4 en el que se observa que la población final obtenida está dominada por el mutante A, que se encuentra acompañado por el AC, que si bien en la mayoría de los casos se encuentra en bajo número, en algunas ocasiones puede presentar un elevado número de efectivos.

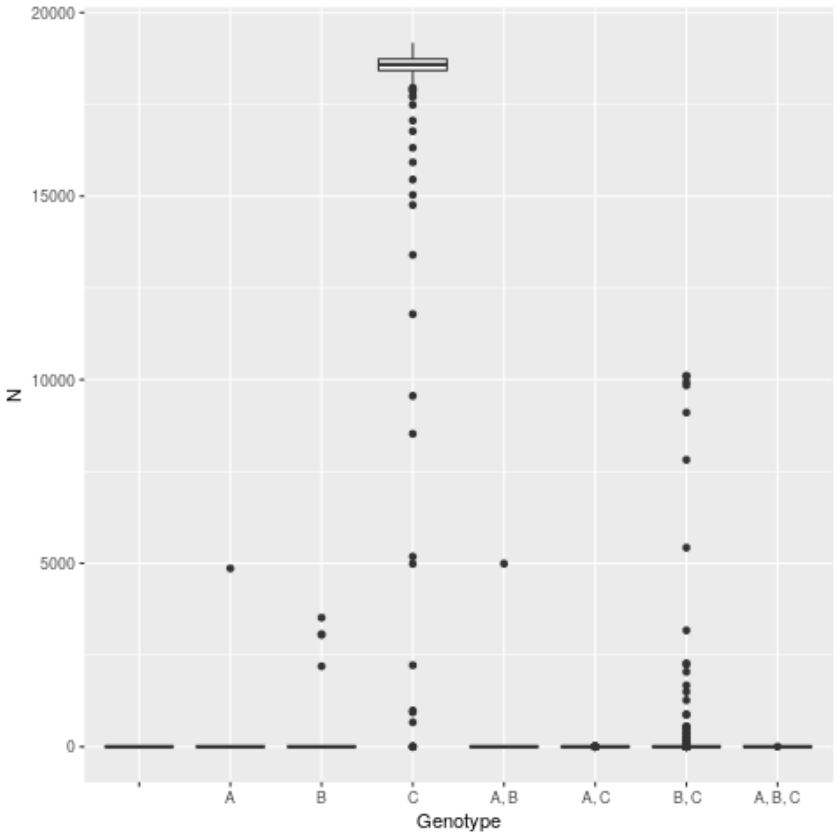


Figura 4.6: Boxplot con la composición poblacional tras correr el caso de Estudio II 1000 veces. En este caso se corrieron las simulaciones utilizando los parámetros de parada por defecto, para disminuir el tiempo de computación, lo cual interrumpe la simulación con anterioridad cuando la población alcanza un determinado tamaño.

Supongamos que a la luz de este conocimiento, encontramos un fármaco específico que es capaz de limitar el crecimiento de todos los subclones que presentan una mutación en A, haciendo que la tasa de natalidad de estos sea la misma que la del WT, es decir, la unidad. Si simulamos

con estas nuevas condiciones, manteniendo el resto de parámetros igual que en la Code box 4.2, se obtienen los resultados que se muestran en la Tabla 4.2 y la Figura 4.5.a.

En esta situación observamos (Figura 4.5.a y Tabla 4.3) que la población final está dominada por el mutante C, que es el que produce el factor proliferativo. Es cierto que a partir de este aparecen los dobles mutantes AC y BC (Figura 4.5.b) que pronto desaparecen debido a que la alta frecuencia de C, implica una alta densidad poblacional, lo que conlleva a que la tasa de mortalidad sea mayor que la de natalidad y estos mutantes desaparecen. Bajo el efecto del fármaco contra los mutantes A, la mayor tasa de mutación del locus C y la producción por parte de este del factor proliferativo parecen dominar la dinámica poblacional.

Los resultados de la composición final de la población tras correr la simulación 1000 veces se encuentran en la Figura 4.6. A la luz de estos resultados, podemos concluir que fármaco es capaz de alterar la dinámica evolutiva, llevándola a un estado en donde el clon C domina la composición poblacional final debido a lo comentado en el párrafo anterior.

5

Conclusiones y Trabajo Futuro

5.1. Conclusiones

A modo de conclusiones que se deriven de la ejecución de este trabajo y de los resultados obtenidos, podemos citar las siguientes:

- La librería ExprTk (Partow, 2018) se comporta de la manera deseada, mostrándose su empleo como una forma flexible de pasar la ecuaciones de fitness al código de C++. Con ella podemos especificar ecuaciones con una amplio abanico de funciones matemáticas (max, min, funciones trigonométricas, etc.), por lo que su adaptación al uso en OncoSimulR (Diaz-Uriarte, 2017) la convierte en pieza clave de la nueva funcionalidad implementada.

- La nueva funcionalidad implementada se comporta como una herramienta de simulación muy flexible, con una gran capacidad para afrontar problemas diversos. Los casos de estudio mostrados en este trabajo vienen a cimentar esta idea, habiéndose demostrado mediante el ejemplo la versatilidad y flexibilidad de la nueva funcionalidad. De este modo, las novedades aportan al paquete OncoSimulR (Diaz-Uriarte, 2017) un mayor rango de escenarios que pueden ser simulados, convirtiendo a este simulador en uno de los más interesantes de los disponibles en la actualidad para trabajar con poblaciones tumorales.

- La nueva funcionalidad implementada está integrada totalmente en el código anterior, como demuestra que pasa el 100 % de los test presentes en el paquete, y es un producto terminado y operativo, a la espera de las comprobaciones pertinentes por parte del autor y director de este trabajo para hacer el pull request y el consiguiente merge con el repositorio principal del proyecto.

5.2. Trabajo Futuro

Los trabajos académicos nunca se terminan, más bien se abandonan llegado el momento. Este no es ninguna excepción. Gran cantidad de aspectos se podrían mejorar y ampliar, entre los que destacamos los siguientes:

- Modificar el código para permitir al usuario simular las situaciones de fitness dependiente de la frecuencia genotípica empleando como variables los tamaños subpoblacionales (frecuencias absolutas), además de las frecuencias relativas. Con ello, se aumentaría la flexibilidad en la especificación de las ecuaciones de fitness, aumentando los escenarios que podrían simularse.

- Permitir al usuario el uso del nombres de genes más allá de las letras y los números en la especificación del fitness, como si se puede realizar cuando no estamos en el caso de fitness dependiente de la frecuencia.

- Realizar simulaciones masivas en distintos escenarios y análisis ulterior de los resultados para estudiar el comportamiento de las nuevas funcionalidades en un marco más amplio. Un ejemplo de estos nuevos escenarios que se podrían explorar es la influencia de realizar simulaciones de fitness dependiente de la frecuencia en las inferencias de las restricciones en el orden de aparición de mutaciones realizadas por los CPMs (Modelos de Progresión de Cáncer). Otro ejemplo de análisis ulterior podría ser la comparación de simulaciones sin el el fitness dependiente de la frecuencia con simulaciones en las que si se tuviera en cuenta esta circunstancia.

- Modificar la forma en que la librería de C++ ExprTk (Partow, 2018) esta incluida en el paquete OncoSimulR (Diaz-Uriarte, 2017) para que este ocupe menos memoria. Una posible linea de mejora sería implementar un archivo *config* que verificara si la librería está presente y disponible en la máquina antes de la instalación del paquete. Si está disponible se realizarían los enlaces pertinentes para que se pudiera emplear en el paquete, descargándola de GitHub o la web del autor y enlazándola, en caso contrario.

Glosario de Acrónimos y Abreviaturas

- **BNB**: Binomial Negative Binomial (Binomial Binomial Negativa). Algoritmo para simular la dinámica evolutiva propuesto por Mather et al. (2012) y cuyo nombre hace alusión a las dos distribuciones probabilísticas que se muestrean en la actualización de las poblaciones.
- **CPM**: Cancer Progression Model (Modelos de Progresión de Cáncer). Estos modelos tratan de identificar las restricciones en el orden de aparición de las mutaciones a lo largo de la progresión tumoral a partir de datos transversales, esto es, datos de diferentes sujetos en el mismo punto temporal, o no considerando diferencias debidas al tiempo. Entre estos CPMs destacamos CBN (Gerstung et al., 2009), CAPRI (Ramazzotti et al., 2015) y OT (Szabo and Boucher, 2008).
- **ej.:** ejemplo.
- **IDE**: Integrated Development Environment (Entorno de Desarrollo Integrado).
- **MIT**: Massachusetts Intitute of Technology (Instituto de Tecnología de Massachusetts).
- **PDF**: Portable Document Format (formato de documento portable).
- **SSA**: Stochastic Simulation Algorithm (Algoritmo de Simulación Estocástica). Algoritmo propuesto por Gillespie (1977) en su origen para simular reacciones químicas acopladas, muy empleado también en la simulación de dinámicas evolutivas.
- **WT**: genotipo Wild Type.

Bibliografía

R. Axelrod, D.E. Axelrod, and K.J. Pienta. Evolution of cooperation among tumor cells. *Proceedings of the National Academy of Sciences of the United States of America*, 103(36): 13474–13479, 2006. https://doi.org/10.1073/pnas.0606053103.

N. Beerenwinkel, N. Eriksson, and B. Sturmfels. Conjunctive bayesian networks. *Bernoulli*, 13 (4):893–909, 2007. https://doi.org/10.3150/07-BEJ6133.

I. Bozic, T. Antal, H. Ohtsuki, H. Carter, D. Kim, S. Chen, R. Karchin, K.W. Kinzler, B. Vogelstein, and M.A. Nowak. Accumulation of driver and passenger mutations during tumor progression. *Proceedings of the National Academy of Sciences*, 107(43):18545–18550, 2010. https://doi.org/10.1073/pnas.1010978107.

A.S. Cleary, T.L. Leonard, S.A. Gestl, and E.J. Gunther. Tumour cell heterogeneity maintained by cooperating subclones in wnt-driven mammary cancers. *Nature*, 508(1):113–117, 2014. https://doi.org/10.1038/nature13187.

R.S. Datta, A. Gutteridge, C. Swanton, C.C. Maley, and T.A. Graham. Modelling the evolution of genetic instability during tumour progression. *Evolutionary Applications*, 6(1):20–33, 2013. https://doi.org/10.1111/eva.12024.

R. Diaz-Uriarte. OncoSimulR: Genetic simulation with arbitrary epistasis and mutator genes in asexual populations. *Bioinformatics*, 33(12):1898–1899, 2017. https://doi.org/10.1093/bioinformatics/btx077.

Eclipse Fundation. *Eclipse: Integrated Development Environment. V Photon 4.8.* Eclipse Fundation, Inc., Ottawa, Ontario, 2018. URL https://www.eclipse.org/.

D. Eddelbuettel. *Seamless R and C++ Integration with Rcpp.* Springer, New York, 2013. URL https://doi.org/10.1007/978-1-4614-6868-4. ISBN 978-1-4614-6867-7.

D. Eddelbuettel and J.J. Balamuta. Extending extitR with extitC++: A Brief Introduction to extitRcpp. *PeerJ Preprints*, 5:e3188v1, aug 2017. https://doi.org/10.7287/peerj.preprints.3188v1.

D. Eddelbuettel and R. François. Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, 40(8):1–18, 2011. https://doi.org/10.18637/jss.v040.i08.

M. Gerlinger, A.J. Rowan, S. Horswell, J. Larkin, D. Endesfelder, E. Gronroos, P. Martinez, N. Matthews, A. Stewart, P. Tarpey, I. Varela, B. Phillimore, S. Begum, N.Q. McDonald, A. Butler, D. Jones, K. Raine, C. Latimer, C.R. Santos, M. Nohadani, A.C. Eklund, B. Spencer-Dene, G. Clark, L. Pickering, G. Stamp, M. Gore, Z. Szallasi, J. Downward, P.A. Futreal, and C. Swanton. Intratumor heterogeneity and branched evolution revealed by multiregion sequencing. *New England Journal of Medicine*, 366(10):883–892, 2012. https://doi.org/10.1056/NEJMoa1113205.

M. Gerstung, M. Baudis, H. Moch, and N. Beerenwinkel. Quantifying cancer progression with conjunctive bayesian networks. *Bioinformatics*, 25(21):2809–2815, 2009. https://doi.org/10.1093/bioinformatics/btp505.

D.T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The journal of physical chemistry*, 81(25):2340–2361, 1977. https://doi.org/10.1021/j100540a008.

J.H. Gillespie. Substitution processes in molecular evolution. I. Uniform and clustered substitutions in a haploid model. *Genetics*, 134(3):971–981, 1993. http://www.genetics.org/content/134/3/971.full.pdf.

GitHub. *Atom: text and source code editor. V 1.29.* GitHub, Inc., San Francisco, CA, 2018a. URL https://atom.io/.

GitHub. *GitHub: web-based hosting service for version control using Git.* GitHub, Inc., San Francisco, CA, 2018b. URL https://github.com/.

M. Greaves and C.C. Maley. Clonal evolution in cancer. *Nature*, 481(7381):306–313, 2012. https://doi.org/10.1038/nature10762.

D. Hanahan and R.A. Weinberg. The hallmarks of cancer. *Cell*, 100(1):57–70, 2000. https://doi.org/10.1016/S0092-8674(00)81683-9.

D. Hanahan and R.A. Weinberg. Hallmarks of cancer: The next generation. *Cell*, 144(5):646–674, 2011. https://doi.org/10.1016/j.cell.2011.02.013.

K.S. Korolev, J.B. Xavier, and J. Gore. Turning ecology and evolution against cancer. *Nature Reviews Cancer*, 14(5):371–380, 2014. https://doi.org/10.1038/nrc3712.

M.C. Lloyd, J.J. Cunningham, M.M. Bui, R.J. Gillies, J.S. Brown, and R.A. Gatenby. Darwinian dynamics of intratumoral heterogeneity: Not solely random mutations but also variable environmental selection forces. *Cancer Research*, 76(11):3136–3144, 2016. https://doi.org/10.1158/0008-5472.CAN-15-2962.

A. Marusyk, D.P. Tabassum, P.M. Altrock, V. Almendro, F. Michor, and K. Polyak. Non-cell-autonomous driving of tumour growth supports sub-clonal heterogeneity. *Nature*, 514(7520):54–58, 2014. https://doi.org/10.1038/nature13556.

W.H. Mather, J. Hasty, and L.S. Tsimring. Fast stochastic algorithm for simulating evolutionary population dynamics. *Bioinformatics*, 28(9):1230–1238, 2012. https://doi.org/10.1093/bioinformatics/bts130.

C.D. McFarland. *The role of deleterious passengers in cancer*. PhD thesis, Harvard University, 2014. http://nrs.harvard.edu/urn-3:HUL.InstRepos:13070047.

C.D. McFarland, K.S. Korolev, G.V. Kryukov, S.R. Sunyaev, and L.A. Mirny. Impact of deleterious passenger mutations on cancer progression. *Proceedings of the National Academy of Sciences*, 110(8):2910–2915, 2013. https://doi.org/10.1073/pnas.1213968110.

L.M.F. Merlo, J.W. Pepper, B.J. Reid, and C.C. Maley. Cancer as an evolutionary and ecological process. *Nature Reviews Cancer*, 6(12):924–935, 2006. https://doi.org/10.1038/nrc2013.

P.C. Nowell. The clonal evolution of tumor cell populations. *Science*, 194(4260):23–28, 1976. https://doi.org/10.1126/science.959840.

J. Ohtsuka, H. Oshima, I. Ezawa, R. Abe, M. Oshima, and R. Ohki. Functional loss of p53 cooperates with the in vivo microenvironment to promote malignant progression of gastric cancers. *Scientific Reports*, 8(1), 2018. https://doi.org/10.1038/s41598-018-20572-1.

A. Partow. Exprtk: Mathematical expression toolkit library, 2018. URL http://www.partow.net/. Open Source MIT License.

B. Peng, M. Kimmel, and C.I Amos. *Forward-time population genetics simulations: methods, implementation, and applications*. John Wiley & Sons, Hoboken, New Jersey, 2012. DOI 10.1002/9781118180358.

J.W. Pepper, C.S. Findlay, R. Kassen, S.L. Spencer, and C.C. Maley. Cancer research meets evolutionary biology. *Evolutionary Applications*, 2(1):62–70, 2009. https://doi.org/10.1111/j.1752-4571.2008.00063.x.

R Core Team. *R: A Language and Environment for Statistical Computing. V 3.5.1*. R Foundation for Statistical Computing, Vienna, Austria, 2018. URL https://www.R-project.org/.

D. Ramazzotti, G. Caravagna, L. Olde Loohuis, A. Graudenzi, I. Korsunsky, G. Mauri, M. Antoniotti, and B. Mishra. Capri: efficient inference of cancer progression models from cross-sectional data. *Bioinformatics*, 31(18):3016–3026, 2015. https://doi.org/10.1093/bioinformatics/btv296.

J.G. Reiter, I. Bozic, K. Chatterjee, and M.A. Nowak. TTP: Tool for tumor progression. In N. Sharygina and H. Veith, editors, *Computer Aided Verification*, pages 101–106, Berlin, Heidelberg, 2013. Springer. https://doi.org/10.1007/978-3-642-39799-8.

RStudio Team. *RStudio: Integrated Development Environment for R. V 1.1.453*. RStudio, Inc., Boston, MA, 2016. URL http://www.rstudio.com/.

A. Szabo and K.M. Boucher. Oncogenetic trees. In W.T. Tan and L. Hanin, editors, *Handbook of Cancer Models with Applications*, pages 1–24. World Scientific, 2008. https://https://doi.org/10.1142/6677.

K.R. Thornton. A C++ template library for efficient forward-time population genetic simulation of large populations. *Genetics*, 198(1):157–166, 2014. https://doi.org/10.1534/genetics.114.165019.

H. Wickham. testthat: Get started with testing. v 2.0. *The R Journal*, 3:5–10, 2011. URL https://journal.r-project.org/archive/2011-1/RJournal_2011-1_Wickham.pdf.

F. Zanini and R.A. Neher. Ffpopsim: an efficient forward simulation package for the evolution of large populations. *Bioinformatics*, 28(24):3332–3333, 2012. https://doi.org/10.1093/bioinformatics/bts633.

A

Material Suplementario

En el presente anexo se pueden encontrar las direcciones url para acceder al material suplementario de este trabajo, así como las instrucciones para la instalación del paquete OncoSimulR (Diaz-Uriarte, 2017) en la versión desarrollada en este trabajo.

El repositorio de GitHub con la versión del paquete implementada para este trabajo (2.10.1) se puede encontrar en https://github.com/sersancar/Oncosimul_fdf.git.

En este repositorio se pueden encontrar las instrucciones necesarias para la instalación del paquete en el fichero README.md, en concreto en la sección Installation. Para una instalación rápida desde el repositorio de GitHub basta con seguir las instrucciones del siguiente Code Box:

```
1 install.packages("devtools") ## if you don't have it already
2 library(devtools)
3 install_github("/sersancar/oncosimul_fdf/OncoSimulR")
```

Code Box A.1: Instalación del OncoSimulR 2.10.1.

En el repositorio se encuentran tanto el paquete en sí, como la documentación y las vignettes. En cualquier caso, la documentación en PDF se puede encontrar en https://github.com/sersancar/TFM_documentation.git y las vignettes tanto en PDF como en html en https://github.com/sersancar/TFM_vignettes.git.

Los ficheros de testeo implementados en este trabajo (*test.allFitnessEffectsFDF.R*, *test.evaluatingGenotypesFDF.R* y *test.Z-oncoSimulIndivFDF.R*) se pueden encontrar en el repositorio citado con anterioridad, en el directorio *OncoSimul/OncoSimulR/test/testthat*.

La librería de C++ ExprTk (Partow, 2018) y su documentación se pueden encontrar en http://www.partow.net/programming/exprtk/.