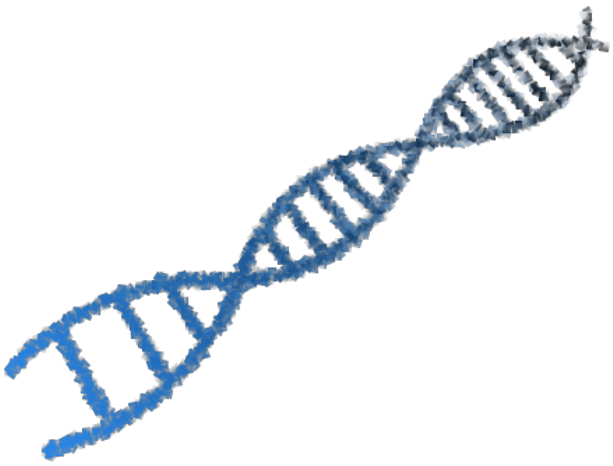
UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Escuela de Ingeniería Informática

Automatización de los procesos de alineamiento y búsqueda de variantes en secuencias de ADN



Pascual Lorente Arencibia

2012/2013

TUTORES

Francisca Quintana Domíguez

Antonio Tugores Cester

Memoria del Proyecto de Fin de Carrera de Ingeniería Informática en la Escuela de Ingeniería Informática de la Universidad de Las Palmas de Gran Canaria.

TÍTULO

Automatización de los procesos de alineamiento y búsqueda de variantes en secuencias de ADN

AUTOR

Pascual Lorente Arencibia

TUTORES

Antonio Tugores Cester. *Jefe de Servicio de la Unidad de Investigación del Complejo Hospitalario Universitario Insular-Materno Infantil*

Francisca Quintana Domínguez. *Profesora Titular de Universidad del Departamento de Informática y Sistemas de la Universidad de Las Palmas de Gran Canaria*

FECHA

noviembre de 2013

RESUMEN

Gracias a la secuenciación de ADN de nueva generación (NGS) es posible obtener grandes cantidades de datos genéticos acerca de un individuo. Las variantes genéticas individuales pueden determinar la presencia de enfermedades genéticas de etiología desconocida. También es posible predecir la susceptibilidad de responder de forma adecuada a un medicamento determinado. El diagnóstico genético supone una mejora en la calidad de los servicios sanitarios del país y las herramientas que existen hoy en día para su análisis están dispersas y son, en muchos casos, hostiles.

Con el desarrollo de este Proyecto de Fin de Carrera en la Unidad de Investigación del Complejo Hospitalario Universitario Insular-Materno Infantil (UICHUIMI) se ha creado una aplicación de escritorio, DNAnalytics, que encapsula los distintos procesos para el análisis de los datos genéticos, aumentando el rendimiento de esta etapa y permitiendo al personal de la unidad conocer de manera más rápida las variantes candidatas de una enfermedad.

AGRADECIMIENTOS

Quiero dar las gracias a todas las personas que han colaborado en el desarrollo de este proyecto:

A mis tutores, a Antonio Tugores por todos sus conocimientos, sin los cuales no habría escrito ni la mitad de este documento, y a Francisca Quintana por su incondicional apoyo y ayuda en la publicación y corrección del mismo.

A todos los compañeros de la UICHUIMI, por los buenos ratos que hemos pasado y que han hecho que este proyecto fuera algo más que una asignatura.

A mis compañeros de clase porque, gracias a ellos y a los buenos momentos vividos juntos, he terminado la carrera como mejor persona.

A mis profesores en la carrera, a ellos les debo mis conocimientos y mis habilidades como profesional.

A mis hermanos Fco. Javier, María, Tomás, José, Judit y Luis, ya que jamás habría llegado tan lejos sin su confianza y apoyo. En especial a Judit, por sus correcciones a la memoria.

A mi novia, Naira, por sus referencias bibliográficas y porque ha sufrido y disfrutado conmigo cada página de este proyecto.

A mis padres, José Luis y July, ya que no estaría donde estoy si no es por todos sus esfuerzos en darme la mejor educación posible y apoyarme, económica y emocionalmente, en todos los proyectos que he emprendido.

Muchas gracias a todos.

Pascual Lorente Arencibia

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Parte I

ANTECEDENTES

Donde se exponen las bases del proyecto, se analiza el estado actual de la UICHUIMI y se formalizan los objetivos.

1

INTRODUCCIÓN

1.1 EL LIBRO DE LA VIDA

El ácido desoxirribonucleico (ADN) almacena la información necesaria para definir la estructura y el comportamiento de la mayor parte de todo organismo vivo conocido. El conjunto de todo el ADN de un organismo se denomina *genoma*, y existe al menos una copia en todas las células de cada organismo, identificándolos así como seres únicos.

El genoma codifica desde el color de los ojos hasta el tamaño y la forma de los pies, contiene las instrucciones para indicar cómo debe trabajar el corazón, los riñones e incluso el cerebro, ordena a las células cómo y cuándo han de reproducirse, cuánto oxígeno es necesario transportar o qué enzimas se deben segregar en el estómago. A todas estas características se las conoce como el *fenotipo*, y es la representación observable de nuestra información genética, el *genotipo*.

El ADN está formado por dos hebras complementarias que están enrolladas entre sí. Cada hebra es una cadena de bases nitrogenadas, unidades mínimas de información equivalentes a un bit informático. Existen 4 bases codificantes: Adenina, Timina, Citosina y Guanina. Por robustez y seguridad, cada hebra contiene las bases complementarias a la otra: la Adenina se complementa con la Timina, y la Guanina con la Citosina.

En concreto, para el ser humano, este libro de instrucciones se almacena en 46 cadenas de ADN, llamadas *cromosomas*, 23 heredadas del padre y 23 heredadas de la madre. 22 de estas parejas son homólogas, es decir, tienen el mismo tamaño y estructura; la última pareja es homóloga en las mujeres –XX– pero no en los hombres –XY–.

Existen, aproximadamente, entre 20.000 y 25.000 genes, regiones del ADN que codifican proteínas, responsables del fenotipo. Los genes representan solamente un 2 % del genoma humano –véase Figura 1.1–, el resto se considera regiones no codificantes y realiza funciones reguladoras y estructurales. Cada gen tiene de media 3000 pares de bases, pero este número varía mucho. El gen más largo conocido

está formado por 2,4 millones de pares de bases. Casi todos los pares de bases –el 99.9 %– son exactamente iguales en todas las personas. [1].

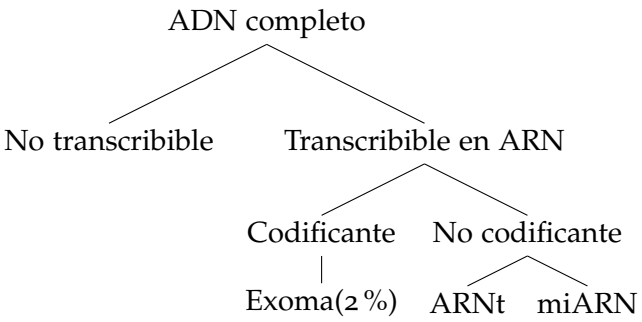


Figura 1.1.: *Estructura del ADN*

Las proteínas son largas y complejas cadenas de aminoácidos, pequeñas moléculas orgánicas que, al unirse, se doblan en estructuras tridimensionales que determinan la función de cada proteína en la célula. Algunas proteínas son hormonas, como la insulina; otras, estructurales, como el colágeno; y otras, funcionales, como la hemoglobina.

El ADN se encuentra en el núcleo de las células, de donde nunca sale. Toda la información es extraída a través del ácido ribonucleico (ARN), una cadena simple de nucleótidos, encargada de transportar y transmitir la información.

El proceso de crear proteínas a partir de ADN se denomina *síntesis de proteínas* que se divide en dos etapas, como muestra la figura 1.2. La primera etapa se conoce como *fase de activación*: el fragmento de ADN que se va a sintetizar es transcrito por el ARN polimerasa (ARNP) en un ARN mensajero (ARNm), que contiene la misma información –complementaria– que el ADN.

El ARNm se desplaza a los ribosomas, unos orgánulos subcelulares, para la segunda fase: la *traducción*. Allí, unas moléculas de ARN de transferencia (ARNt) se emparejan con el ARNm cada tres bases –en el ARNm cada trío es conocido como codón y en el ARNt, anticodón– . Cada ARNt reconoce y se asocia con uno de los 20 aminoácidos presentes en las proteínas. A medida que un ARNt se complementa con el ARNm, su aminoácido se une a la proteína en construcción. Este proceso comienza en el codón de inicio y termina en el codón de terminación, momento en el que la proteína está lista para ser liberada.[21]

1 http://www.ornl.gov/sci/techresources/Human_Genome/project/info.shtml

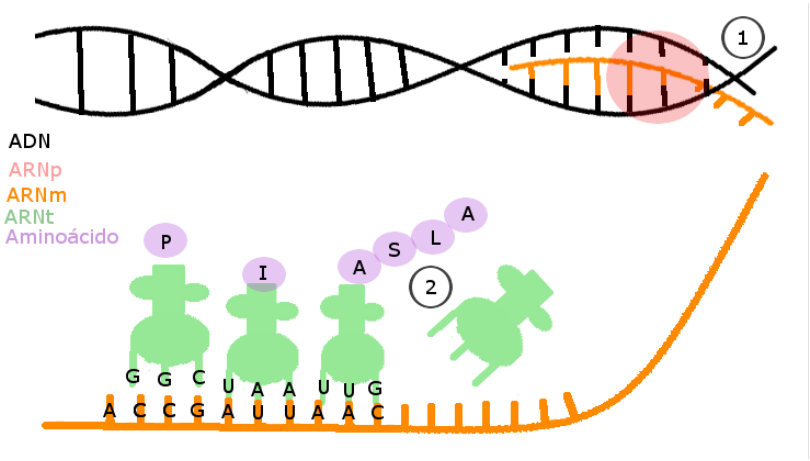


Figura 1.2.: *Síntesis de proteínas. (1) Activación: en el núcleo, el ARNp genera un ARNm transcribiendo el ADN. (2) Traducción: en los ribosomas, el ARNt se va alineando con el ARNm para generar la cadena de aminoácidos.*

Como cada codón está formado por 3 bases y existen 4 bases, hay un total de 64 codones para codificar 20 aminoácidos, representados en la Tabla 1.1. Tres de los codones no codifican ningún aminoácido y se denominan *codones de terminación*, ya que su presencia supone la terminación del proceso de síntesis. El codón de inicio indica el comienzo de la síntesis de proteínas y también codifica para la Metionina.

UUU Fenilalanina	UCU Serina	UAU Tirosina	UGU Cisterina
UUC Fenilalanina	UCC Serina	UAC Tirosina	UGC Cisterina
UUA Leucina	UCA Serina	UAA Codón de fin	UGA Codón de fin
UUG Leucina	UCG Serina	UAG Codón de fin	UGG Triptofán
CUU Leucina	CCU Prolina	CAU Histidina	CGU Arginina
CUC Leucina	CCC Prolina	CAC Histidina	CGC Arginina
CUA Leucina	CCA Prolina	CAA Glutamina	CGA Arginina
CUG Leucina	CCG Prolina	CAG Glutamina	CGG Arginina
AUU Isoleucina	ACU Treonina	AAU Asparagina	AGU Serina
AUC Isoleucina	ACC Treonina	AAC Asparagina	AGC Serina
AUA Isoleucina	ACA Treonina	AAA Lisina	AGA Arginina
AUG Metionina	ACG Treonina	AAG Lisina	AGG Arginina
GUU Valine	GCU Alanina	GAU Ácido aspártico	GGU Glicina
GUC Valine	GCC Alanina	GAC Ácido aspártico	GGC Glicina
GUA Valine	GCA Alanina	GAA Ácido glutámico	GGA Glicina
GUG Valine	GCG Alanina	GAG Ácido glutámico	GGG Glicina

Tabla 1.1.: *Tabla de codones*

1.2 HERENCIA Y EVOLUCIÓN

El ADN de un nuevo individuo se genera en el momento de la fecundación, cuando los gametos –óvulo y espermatozoide– se unen para dar origen a la primera célula madre.

Los óvulos y los espermatozoides contienen solo la mitad de la información genética –23 cromosomas–. Esto se produce con la división celular, la meiosis. En el hombre, de una célula llamada espermatogonia que tiene 23 parejas de cromosomas, se obtienen cuatro espermatozoides con 23 cromosomas cada uno. En la mujer, de una ovogonia –23 parejas– se obtiene un óvulo–23 cromosomas–.

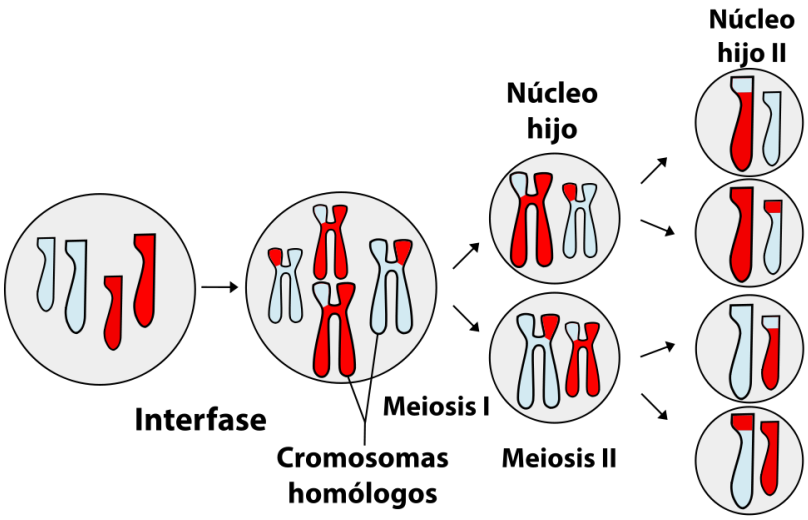


Figura 1.3.: *Visión general de la meiosis. En la interfase se duplica el material genético. En meiosis I se produce el fenómeno de entrecruzamiento y los cromosomas homólogos se reparten en dos células hijas. En meiosis II, al igual que en una mitosis, cada cromátida migra hacia un polo. El resultado son 4 células hijas haploides (n).*

Durante esta división se produce el *entrecruzamiento cromosómico*: cromosomas homólogos se entrecruzan e intercambian fragmentos entre sí, haciendo que cada espermatozoide contenga una secuencia genética distinta [21]. Este mecanismo asegura que el cromosoma que se transmite a la progenie contenga información de ambos cromosomas parentales.

Se estima que el proceso de duplicación del ADN tiene una tasa de error entre 10^{-7} y 10^{-8}, es decir, un error cada 10 a 100 millones de

copias de bases. [23]. Teniendo en cuenta que hay 3 mil millones de pares de bases en un ADN, y que solo un 2 % es codificante:

$$\frac{3x10^{9} bases}{ADN} x \frac{2x10^{-2} ADN}{codificante} x \frac{10^{-7} errores}{base} = 6 \frac{errores}{codificante}$$

Lo cual significa que aparecen menos de 10 variantes espontáneas por individuo. Este número es bastante bajo, porque solo indica las posibles nuevas variaciones en el ADN que introduce un individuo. Además de las variantes generadas en la meiosis, también se heredan las que contienen los progenitores. De hecho, es más probable que un ser humano muestre una variante heredada a que sea espontánea, ya que cada uno de los progenitores ha heredado, a su vez, las de sus progenitores y así sucesivamente, de modo que el ADN de una persona puede definirse como la infinitud de variantes introducidas generación tras generación.

Existen variantes del tamaño de un gen –como la traslocación, en la que dos fragmentos de ADN se intercambian–, del tamaño de un cromosoma –como la trisomía, en la que existen 3 alelos, en lugar de 2, del mismo cromosoma– y variantes puntuales, que afectan a uno o varios pares de bases. Las variantes puntuales se pueden clasificar por su morfología:

- *Single Nucleotide Polymorphism (SNP):* una base es sustituida por otra, sin alterar el tamaño de la cadena. Como un codón es definido por tres bases, es posible que el nuevo codón siga codificando el mismo aminoácido –variante sinónima–, o que haya un cambio en la secuencia de una proteína –no sinónima–.

- *Inserciones/Delecciones (InDels):* una o varias bases son insertadas o eliminadas de un punto concreto de la cadena. Dependiendo de si la InDel es múltiplo de 3, o de la posición en el gen en que se encuentra, puede modificar drásticamente la proteína resultante al alterar la pauta de lectura.

No todas las variantes contenidas en el ADN se muestran en el fenotipo. Gregor Mendel, considerado el padre de la genética moderna, estableció tres leyes para determinar cómo se transmiten las características en cada generación. Para ello, cosechó guisantes hasta obtener lo que él denominaba *razas puras*, razas que por mucho que se cruzaran siempre tendrían las mismas características. En su caso, guisantes de color verde y de color amarillo. Entonces cruzó estas dos razas, llegando a su primera conclusión.

Ley de la uniformidad de los híbridos de la primera generación filial: al cruzar entre sí dos razas puras se obtiene una generación filial que es idéntica entre sí e idéntica a uno de los padres –Figura 1.4–.

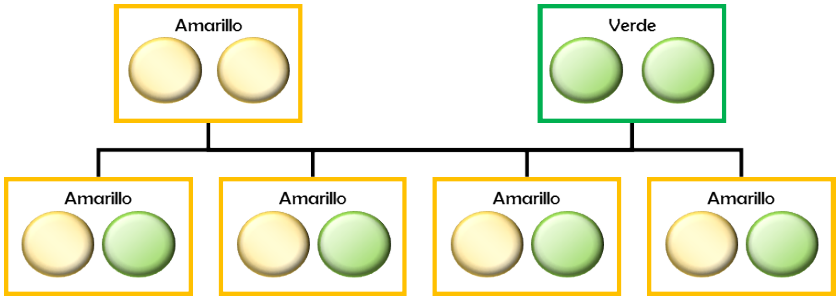


Figura 1.4.: *1ª ley de Mendel.*

Todos los guisantes cosechados eran amarillos, con lo cual dedujo que existen características que son dominantes y características que son recesivas. Los hijos son idénticos al padre que tenga una característica dominante, lo que implica que la característica recesiva desaparece, aunque Mendel no lo creía así. Por eso, volvió a cruzar los guisantes cosechados, a los que llamó *híbridos de primera generación*.

Ley de la segregación de los caracteres en la segunda generación filial: al cruzar entre sí dos híbridos, los factores hereditarios de cada individuo se separan, ya que son independientes, y se combinan entre sí de todas las formas posibles –Figura 1.5–.

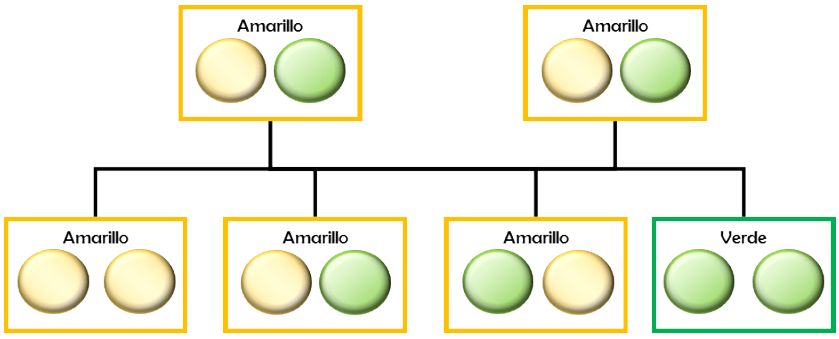


Figura 1.5.: *2ª ley de Mendel.*

En esta segunda generación obtuvo tanto guisantes verdes como guisantes amarillos, en una proporción aproximada de tres amarillos por cada verde. Esto demostró su teoría de que los individuos conservan características de ambos progenitores, y que en cada generación se vuelven a combinar.

Por último, repitió los experimentos, pero esta vez utilizó hasta siete características distintas para observar la dependencia entre ellas: color de la semilla, color de la flor, rugosidad de la semilla, longitud de la vaina, distribución de las flores en el tallo, color de la vaina y rugosidad de la vaina.

> *Ley de la independencia de los caracteres hereditarios*: al cruzar entre sí dos dihíbridos los caracteres hereditarios se separan, ya que son independientes y se combinan entre sí de todas las formas posibles–Figura 1.6–.

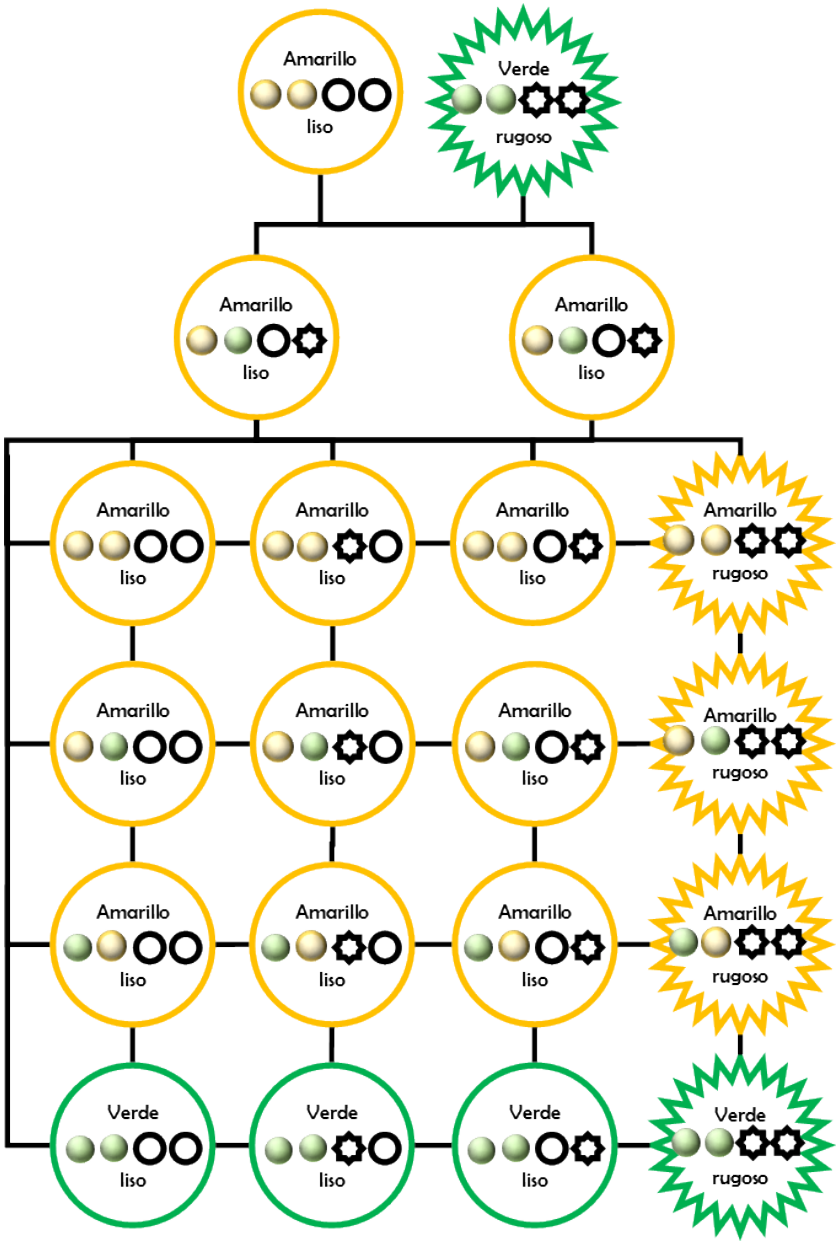


Figura 1.6.: *3ª ley de Mendel.*

Los experimentos de Mendel han servido para explicar el comportamiento de la herencia del fenotipo, así como la presencia de los dos alelos en cada individuo. [16, 18]

Hoy en día sabemos que en realidad no todas las características son independientes entre sí. La dependencia viene determinada por su posición en el ADN. Cuanto más cerca se encuentren, más dependientes serán, ya que existen más posibilidades de que se conserven juntas en el entrecruzamiento. La frecuencia de recombinación es una forma de medir la distancia genética entre dos caracteres, que se mide en centimorgan (cM). 1 centimorgan se define como la distancia entre dos posiciones cromosómicas cuya probabilidad de separarse en el entrecruzamiento es 1 %. De promedio, esta distancia es aproximandamente de 1 millón de bases[21].

Para que una variante se convierta en *mutación*, el efecto en el individuo tiene que ser negativo. En otras palabras, supone una desventaja a la hora de relacionarse con el entorno o reproducirse. Si el efecto es positivo, se denomina *ventaja*. Que una variante sea mutación o ventaja depende del entorno, a esto se le conoce como *presión selectiva*.

Muchas de las variantes en el ADN son toleradas por el cuerpo humano y no representan ningún cambio significativo en el fenotipo. Otras, implican un cambio perceptible en el aspecto físico –como el albinismo– o en algún órgano –como el daltonismo–. Por último, existen variantes genéticas causantes de graves enfermedades, en muchos casos letales. En este grupo se encuentran la neurofibromatosis o la enfermedad de Tay-Sachs, entre otras. Son este tipo de variantes las que se conocen como mutaciones. La base de datos OMIM (NCBI)[8], almacena y publica todas estas patologías. Además, un mismo gen puede causar patologías diferentes y una misma patología, ser causada por distintos genes.

1.3 APLICACIONES MÉDICAS

Descifrar las secuencias de ADN es esencial para muchas ramas de la investigación biológica. Con la llegada de la secuenciación de Sanger basada en la electroforesis capilar (CE) los científicos obtuvieron la capacidad de dilucidar la información genética de cualquier sistema biológico. Más tarde, y debido a las limitaciones en rendimiento, escalabilidad, velocidad y resolución de este método, surgió la secuenciación de nueva generación (NGS), un enfoque distinto a la secuenciación Sanger que está desatando una revolución en la ciencia genómica.

En principio, el concepto detrás de la NGS es similar a Sanger, las bases son secuencialmente identificadas a través de señales emitidas. Mientras que Sanger se limita a secuenciar un fragmento determinado –de cualquier longitud–, NGS amplía este proceso a través de millones de reacciones de forma paralela y aleatoria.

Para ello, cada muestra de ADN genómico (ADNg) es fragmentada en una biblioteca de pequeños segmentos que pueden ser secuenciados de forma uniforme y precisa en millones de reacciones paralelas. Las recién identificadas cadenas de bases, llamadas *lecturas*, son entonces reensambladas utilizando, o no, un genoma de referencia conocido. El conjunto completo de lecturas alineadas revelan la secuencia completa de cada cromosoma en la muestra de ADNg. Si no existe un genoma de referencia para la especie que se está secuenciando, se utiliza un ensamblado *de novo*.

En resumen, NGS consiste en fragmentar varias muestras del mismo ADN en millones de trozos pequeños que son muy fáciles de leer, para luego reconstruirlo a través de un computador. Esta estrategia es posible gracias a la alta potencia computacional disponible.

Una de las aplicaciones médicas de la secuenciación que actualmente está en uso es la farmacogenética, que estudia la respuesta de un paciente con una determinada variante ante distintos medicamientos. La efectividad o toxicidad de un fármaco viene determinada por los genes, lo que significa que un medicamento que es beneficioso para una persona, puede ser inútil o dañino para otra. Esta disciplina ayuda a personalizar los tratamientos médicos, mejorando el uso que se hace de los fármacos.

Otra de las aplicaciones es la de diagnosis de enfermedades no manifestadas. Con una secuenciación de exoma, se puede conocer la presencia de enfermedades graves como la fibrosis quística o la enfer-

medad de Wilson. Esto permite aplicar tratamientos preventivos, que resultan más económicos y mejoran la calidad de vida del paciente.

En último lugar, la secuenciación de exoma permite descubrir el origen genético de una enfermedad, permitiendo el tratamiento de la enfermedad no solo combatiendo los síntomas, sino también entendiendo su origen para dirigir de forma más eficaz su tratamiento.

Hoy en día, el procesamiento y el análisis de tanta información es la principal limitación para acercar esta tecnología a la práctica clínica. Es por ello que en este Proyecto de Fin de Carrera se crea una aplicación para automatizar estos procesos con el objeto de reducir esta limitación y establecer las bases de un trabajo futuro donde la secuenciación de ADN pueda ser un estándar en la práctica clínica.

2

OBJETIVOS

El objetivo principal del proyecto es la creación de una aplicación que permita al usuario buscar, identificar, filtrar y comparar las variantes genéticas que presenta uno o varios individuos.

La finalidad de esta herramienta es aplicar la secuenciación de exoma completo al diagnóstico clínico de enfermedades genéticas de etiología desconocida y a la predicción de la susceptibilidad de responder de forma adecuada a un medicamento determinado.

Este objetivo se alcanza a través de los siguientes subobjetivos:

- Identificación de los procesos necesarios para cada una de las tareas –alineación, búsqueda, filtrado y comparación– que se realizan o se espera realizar en la UICHUIMI.

- Síntesis de estos procesos en flujos transparentes al usuario, limitando el intercambio de información al lenguaje genético.

- Creación de una interfaz gráfica para ejecutar cada uno de los procesos o tareas.

3

ESTUDIOS PREVIOS

3.1 LA UICHUIMI

La Unidad de Investigación del Complejo Hospitalario Universitario Insular-Materno Infantil (UICHUIMI) lleva algunos años ya tratando de descubrir el origen de enfermedades genéticas en la población local con el fin de diagnosticar y proponer un tratamiento adecuado a estas nuevas enfermedades. Uno de sus mayores logros ha sido el de indentificar una mutación endémica en la población afectada por la enfermedad de Wilson en la isla de Gran Canaria[11, 12].

Para identificar las variantes genéticas responsables de varias enfermedades mendelianas no caracterizadas en Gran Canaria, la UICHUIMI trabaja con la secuenciación de exoma por NGS, ya que cubre todo el exoma con lecturas muy fiables de un alto porcentaje del exoma. De las miles de variantes que posee cada ser humano, hay que dilucidar aquellas sospechosas de producir la enfermedad de estudio, lo que conlleva un proceso arduo y, en la mayoría de los casos, manual.

La secuenciación por NGS genera ficheros de datos considerablemente grandes, con decenas de millones de líneas de información. Para su tratamiento existe software específico, disperso y, generalmente, poco intuitivo. Por eso, la UICHUIMI confía en que la integración del conocimiento informático permita converger hacia un proceso estandarizado, atractivo y económicamente viable que pueda integrarse en el sistema sanitario.

3.2 EL PROCESO COMPUTACIONAL GENÉTICO

A continuación se describe el proceso que se sigue desde que se elabora una o varias hipótesis hasta que se obtienen las variantes susceptibles de provocar la enfermedad.

3.2.1 *Elaboración de hipótesis*

En primer lugar, se hace un estudio de la familia del paciente o los pacientes susceptibles de padecer una enfermedad de origen genético. Se elabora un árbol genealógico –como el de la Figura 3.1–, donde se describe quién tiene la enfermedad y qué parentesco guarda con el resto. Entonces, se redactan una o varias hipótesis que cubran todos los casos posibles acerca del tipo de variante genética. Dominante o recesiva y autosómica –si no se encuentra en los cromosomas sexuales– o ligada al sexo.

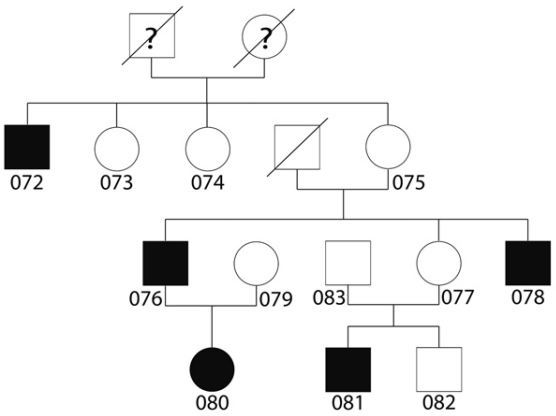


Figura 3.1.: *Árbol genealógico. En negro, los individuos afectados. Los cuadrados representan hombres y los círculos mujeres. Las figuras con interrogantes se desconoce si padecen o no la enfermedad, y las tachadas han fallecido. Los números preservan la identidad de los individuos.*

Por ejemplo, el árbol genealógico de la Figura 3.1 nos ayuda a establecer las distintas hipótesis:

1. Teniendo en cuenta que 80 y 76 están afectados, la variante puede ser heterocigota dominante, es decir, 79 no está afectada, 80 hereda la variante solamente de 76. Entonces, podemos buscar

la variante entre las comunes de 76 y 80, pero no presentes en 79.

2. Si nos fijamos en 81, su madre, 77, no está afectada por la enfermedad, pero como 76 y 78 sí están afectados, es probable que sea asintomática, algo frecuente en enfermedades dominantes. Si solo conociéramos esta parte de la familia, podríamos pensar que la enfermedad es recesiva, lo que implica que 83 y 77 son portadores. Aquí, por ejemplo, podríamos buscar entre las comunes entre 81, 78, 77 y diferentes a 82.

Al establecer las hipótesis se genera la planificación para las fases posteriores, decidiendo qué cadenas de ADN son necesarias secuenciar y cómo se deben comparar los datos obtenidos.

3.2.2 *Obtención del ADN*

Una vez se ha decidido a qué pacientes se les va a secuenciar el ADN, se les extrae una muestra de 10 mililitros sangre. Se utiliza una simplificación del método de precipitación con sales descrito por Miller[22] para aislar el ADN en un tubo de ensayo donde podrá ser conservado.

3.2.3 *Secuenciación del ADN*

Las cadenas aisladas de ADN, conocidas como muestras, se envían a BGI (Beijing Genomics Institute, China[1]), que se encarga de la secuenciación por NGS. Allí, fragmentan el ADN en millones de trozos de entre 150 y 200 pares de bases. Por medio de técnicas físico químicas, se separan los fragmentos de menor calidad y aquellos que no pertenecen al exoma. Finalmente, los fragmentos restantes son secuenciados en ambas direcciones utilizando una máquina llamada HiSeq 2000[2].

El resultado de la secuenciación del ADN es almacenado en archivos informáticos que contienen todas las lecturas generadas junto a un valor de calidad. Las lecturas recibidas están contenidas en dos ficheros FASTQ A.2, cada uno representando todas las secuencias leídas en ambas direcciones, así como un valor de calidad para cada lectura.

1 http://www.genomics.cn/en/index
2 http://www.genomics.cn/en/navigation/show_navigation?nid=4145

3.2.4 *Alineamiento de las secuencias*

Los datos recibidos de la secuenciación se encuentran en formato FASTQ (ver apéndice A.2), en dos ficheros distintos. Estos ficheros suelen contener una media de 30 millones de secuencias de 80 a 100 pares de bases, lo que hace un total de 3 mil millones de pares de bases secuenciados.

El formato FASTQ no aporta información acerca de en qué posición se encuentran las secuencias y ese es el primer paso que hay que dar, alinear cada secuencia con su posición correcta o, al menos, con la más probablemente correcta. Para ello, cada secuencia se compara con un genoma de referencia, el cual es publicado por el Genome Reference Consortium[1] en formato FASTA –ver A.1–. La versión actual de este genoma es la GRCh37 (patch 13, 28 de junio de 2013).

El alineamiento del genoma es un problema de comparación de cadenas a gran escala. El alto coste en cómputo y memoria hacen imposible utilizar algoritmos tradicionales y hay que utilizar métodos que sacrifiquen eficacia para aportar mayor eficiencia. Se pueden diferenciar dos tipos de algoritmos para resolver este problema:

1. *Algoritmos basados en tablas hash:* Se utiliza una pequeña tabla hash para almacenar el índice de las secuencias o del genoma de referencia. La principal ventaja de este método es que requiere poca memoria para funcionar. La variedad de estos algoritmos radica en el tipo de semillas utilizadas y en la resolución de conflictos.

2. *Algoritmos basados en árboles sufijo/prefijo:* Se almacenan las secuencias del genoma de referencia en un árbol, de modo que es muy fácil localizar las lecturas. Además, el alineamiento de varias copias iguales del mismo fragmento se hace una sola vez. La mayor desventaja de este método es que no hay memoria suficiente para almacenar esos árboles, por lo que se utilizan árboles incompletos, los sufijo/prefijo, que solo almacenan parte de las secuencias. Además, esto requiere indexar el genoma de referencia antes de empezar el alineamiento.

Burrows-Wheeler Aligner

BWA[14] es un paquete de programas software para mapear secuencias cortas respecto a un gran genoma de referencia, como, por ejemplo, el genoma humano. Está formado por tres algoritmos: BWA-

backtrack, BWA-SW y BWA-MEM. El primer algoritmo está diseñado para lecturas de secuencias de hasta 100 pares de bases, mientras que los otros dos están diseñados para secuencias más largas, desde 70 hasta 1 millón de pares de bases. BWA-SW y BWA-MEM comparten características similares tales como soporte para lecturas largas y alineamiento de rupturas pero BWA-MEM, que es el último, se recomienda para consultas de alta calidad, ya que es más rápido y preciso. BWA-MEM también tiene mayor rendimiento que BWA-backtrack para lecturas Illumina de 70 a 100 pares de bases. Para cualquier algortimo, BWA necesita construir un índice de tipo FM-index –un tipo de árbol sufijo/prefijo– para el genoma de referencia, aunque solo es necesario hacerlo una vez.

BWA-back está diseñado principalmente para tasas de error de secuenciación por debajo del 2 %. Aunque se le puede permitir tolerar más errores por línea de comando, su rendimiento se degrada rápidamente. Para lecturas de Illumina, bwa-backtrack puede eliminar bases de baja calidad del alelo 3' antes de alinear y, así, alinear más lecturas por la cola, que es una situación típica de Illumina.

BWA-SW y BWA-MEM toleran más errores con alineamientos más largos. Las simulaciones sugieren que pueden trabajar bien con un 2 % de error para alineamientos de 100 pares de bases (100bp), 3 % para 200bp, 5 % para 500bp y 10 % para 1000bp o más.

La salida de los algoritmos BWA está en formato SAMA.3, que es compatible con varios buscadores de variantes.

Bowtie 2

Bowtie 2[13] es una herramienta ultrarrápida y eficiente para alinear lecturas de secuencias respecto a un gran genoma de referencia. Es particularmente buena alineando lecturas de entre 50bp y 100bps con grandes genomas. De hecho, no hay límite en el tamaño de las lecturas. Bowtie 2 indexa el genoma con un FM-index para mantener un bajo consumo de memoria: para el genoma humano, el consumo es de alrededor de 3.2 GB de RAM. Bowtie 2 soporta alineamiento local, con huecos y de doble cadena. Se pueden utilizar múltiples procesadores en paralelo para aumentar la velocidad.

El formato de salida de Bowtie 2 es SAM.

Short Oligonucleotide Analysis Package

SOAP[15] ha ido evolucionando desde una sencilla herramienta de alineamiento a un paquete que proporciona soluciones para el análisis de datos NGS.

Actualmente consiste en una herramienta de alineado (SOAPaligner/soap2), un buscador de variantes estructurales (SOAPsnp) y un buscador de inserciones/deleciones (SOAPindel) entre otras. Incluso están implementando una herramienta de alineamiento acelerada por GPU.

SOAPaligner/soap2 es un alineador muy rápido y preciso para enormes cantidades de lecturas cortas generadas por Illumina/Solexa Genetic Analyzer. Comparado con su versión anterior (soap v1), es un orden de magnitud más rápido. Tan solo requiere dos minutos para alinear un millón de secuencias con el genoma humano. Otra mejora de SOAPAligner es que ahora soporta un amplio rango de longitud de lecturas.

SOAPAligner mejora el uso del tiempo y el espacio gracias a una revolución en las estructuras básicas de datos y algoritmos utilizados. Los algoritmos del núcleo y la estructura de índice (2way-BWT) son desarrollados por el grupo de investigación del Departamento de Informática de la Universidad de Hong Kong (T.W. Lam, Alan Tam, Simon Wong, Edward Wu and S.M. Yiu).

La última versión de SOAPAligner data del año 2011, aunque el paquete SOAP sigue activo, y el formato de salida es propio e incompatible con otros programas.

3.2.5 *Localización de variantes*

El siguiente paso consister en ir desde el principio hasta el final del genoma de referencia, comparándolo con el exoma reconstruido, e identificando en qué posiciones es diferente. Este proceso es el más complejo de todos, ya que en cada posición suele haber varias lecturas alineadas y pueden diferir, y hay que tener en cuenta que hay variantes en las que falta o sobra un fragmento –InDels–.

Antes de proceder a la búsqueda de variantes, se refinan los alineamientos. Utilizando *The Genome Analysis Toolkit* (GATK[3]) se realinean los vacíos en las lecturas para facilitar una detección de InDels

robusta, se eliminan lecturas repetidas que no añaden información y se recalibra con un modelo estadístico de los valores de calidad.

Un ejemplo de cómo se encuentran las secuencias en un fichero SAM es el siguiente:

```
 ref : TCGTAAGTAGTCA
seq1 : ---TACGTAG---
seq2 : TCGTAC-------
seq3 : -----CGTAGTCA
seq4 : TCGTACG------
seq5 : -----AGTAGTCA
```

En la posición 6 de la referencia hay una Adenina, sin embargo, en 4 de las 5 secuencias hay una Citosina. Aunque pueda parecer evidente que existe una variación en esa posición, hay que tener en cuenta algunos aspectos:. uno, que cada lectura tiene un valor de calidad, por lo que esta variación puede ser debida a un error en la máquina secuenciadora; o dos, que cada alineamiento tiene un valor de bondad, por lo que puede ser un fallo del programa alineador.

Es cuestión del programa localizador de variantes dar con el mayor número de verdaderos positivos. Además, este es el caso más sencillo de variación: existen variaciones por la ausencia de una o más bases, por la repetición de un trozo de cadena o por la inserción de otra cadena.

Para determinar la bondad de una variante se establece un modelo probabilístico, siendo muy común usar la inferencia bayesiana[17], una aproximación subjetiva que va mejorando a medida que se añaden evidencias.

The Genome Analysis Toolkit

GATK[3] es un paquete de herramientas desarrollado en el Broad Institute (Harvard, MA, EEUU), una institución sin ánimo de lucro dedicada íntegramente al estudio genético, para analizar datos NGS[10]. El paquete ofrece una amplia gama de herramientas, con un enfoque primario en la búsqueda de variantes y el genotipado, así como un fuerte énfasis en el control de calidad. Su robusta arquitectura, su potente motor de procesamiento y sus características de computación de alto rendimiento lo hacen capaz de manejar proyectos de cualquier tamaño.

Debido a su arquitectura, GATK es muy genérico y puede ser aplicado a todo tipo de bases de datos y problemas de análisis de genoma. Se puede utilizar tanto para búsqueda como validación. Maneja igualmente exomas y genomas completos. Puede trabajar con información generada con distintas tecnologías de secuenciamiento. Y, aunque en un principio se desarrolló para genética humana, GATK ha evolucionado para manejar información de cualquier organismo, con cualquier nivel de ploidía.

Para la búsqueda de variantes proporciona la herramienta HaplotypeCaller, que localiza tanto InDels como SNPs usando un ensamblado de-novo de haplotipos. Los haplotipos son evaluados con un sitema estadístico denominado errores afines a huecos por Pares HMM. En otras palabras, recorre la referencia y allí donde encuentra una variante, aplica las fórmulas del modelo estadístico, obteniendo una bondad en base a la cual determina si es o no una variante real. El modelo se retroalimenta, por lo que a medida que van apareciendo más variantes, se obtienen mejores resultados.

Dindel

Dindel[5] es un programa para localizar pequeños indels en datos NGS que realinea las lecturas en haplotipos candidatos. La manera más simple de utilizar Dindel es considerar todos los indels de un fichero BAM, y comprobar cuáles de ellos son verdaderos indels y cuáles errores de secuenciación o mapeo.

Dindel también puede comprobar indels candidatos de otras fuentes, como una base de datos de variantes conocidas, o indels localizadas de distintas muestras, para saber si también están presentes en las secuencias de la muestra actual. Solo trabaja con secuencias generadas por por la plataforma Illumina GA, ya que utiliza un modelo de error específico.

3.2.6 *Anotación de variantes*

Tener un fichero con cientos de miles de variantes no es útil cuando el objetivo es localizar unas pocas entre ellas. Para ello, es necesario filtrar en base a ciertos criterios. Como la localización solo especifica en qué lugar se produce una variación y cuál es, es necesario añadir más información a cada variante.

Los campos más comunes que se añaden son:

- *Localización:* Estos campos indican a qué codón pertenece la variante y qué codón había en el genoma de referencia. Además, se pueden incluir campos que especifiquen la proteína que codifica y el exón/intrón y el gen donde se encuentra.

- *Sinonimia:* Una variante es sinónima cuando no representa un cambio en el aminoácido que codifica el codón donde se encuentra, ya que se sigue transcribiendo la misma proteína.

- *Peligro:* Dentro de las variantes no sinónimas, existen las toleradas y las peligrosas, dependiendo de cómo es el cambio en la variante.

- *Conocimiento:* Lista de bases de datos y publicaciones donde se encuentra la variante, lo que nos dice hasta qué punto es conocida la variante.

- *Frecuencia:* Frecuencia de la variante en las distintas bases de datos consultadas. Normalmente se clasifican por regiones, de modo que se puede saber en qué lugares del mundo es más o menos frecuente. La base de datos genómica más popular es la de los 1000 Genomas[6].

Aunque muchas de las herramientas de anotación permiten ejecutarse localmente, su potencial reside en el servicio web, ya que los resultados siempre están actualizados.

Sorting Intolerant From Tolerant

SIFT [19] predice si la sustitución de un aminoácido afecta a la función de una proteína basándose en su homología con otras secuencias y las propiedades físicas del aminoácido. SIFT se puede aplicar sobre polimorfismos no sinónimos naturales o mutaciones sin sentido inducidas en laboratorio. Se centra en variantes localizadas en zonas codificantes del ADN.

Las anotaciones de SIFT son las siguientes: nombre y descripción del gen, identificación, descripción y tamaño de la proteína, estado de la transcripción en Ensembl –conocida o nueva–, consecuencias –dañina, tolerada o desconocida–, similitudes con ratones y macacos, enfermedad en OMIM[8] –catálogo online de desórdenes genéticos–, frecuencias de Hapmap –población completa y por continentes–.

Variant Effect Predictor

Ensembl provee la facilidad de predecir las consecuencias funcionales de variantes conocidas y desconocidas usando VEP[25]. Puede identificar variantes en todo el ADN, no solo en regiones codificantes.

Entre las anotaciones cabe destacar: nombre del gen, trasncritos asociados, consecuencias, posición en el ADN codificante (ADNc), posición en CDS, posición en la proteína, cambio de aminoácido, cambio de codón, información de otras herramientas online –SIFT, Polyphen–
.

ANNOVAR

ANNOVAR es una herramienta software eficiente que utiliza información actualizada para anotar variantes genéticas detectadas en distintos genomas –incluyendo genoma humano, de ratón, gusano, mosca, levadura y muchos otros–. Dada una lista de variantes con cromosoma, posición de inicio y de fin, nucleótido de referencia y observado, ANNOVAR puede hacer:

- *Anotación por gen:* Identifica cambios causados en la codificación de la proteína por un SNP y los aminoácidos involucrados. Se pueden utilizar genes de RefSeq, UCSC, ENSEMBL, GENCODE y otros sistemas de definición.

- *Anotación por región:* Identifica variantes en una o varias regiones determinadas, por ejemplo, regiones conservadas en 44 especies, zonas de unión de factores en transcripciones predichas, zonas de duplicación de segmentos, bases de datos de variantes genómicas o muchas otras anotaciones.

- *Anotación por filtros:* Identifica variantes presentes en dbSNP, o identifica el subconjunto de SNPs comunes (MAF $>1\,\%$) en el 1000 Genome Project, o identifica el subconjunto de SNPs no sinónimos con un valor mayor a 0,05 en SIFT, o encuentra variantes intergénicas con un valor de GERP++ menor que 2.

- *Otras funcionalidades:* Recupera la secuencia de nucleótidos en cualquier posición especificada o identifica una lista de genes candidatos para una enfermedad mendeliana.

Polymorphism Phenotyping v2

PolyPhen-2[4] es una herramienta que predice el posible impacto de una sustitución de aminoácido en la estructura y funcionalidad de una proteína humana usando sencillas consideraciones físicas y comparativas.

Polyphen-2 es un nuevo desarrollo de la herramienta Polyphen, algunas de sus mejoras son: flujo de trabajo de alineamientos de múltiples secuencias de alta calidad, clasificador probabilístico basado en métodos de aprendizaje de la máquina y otpimización para el análisis de información NGS de alto rendimiento. Polyphen se concentra en variantes de tipo SNPs no sinónimas en regiones codificantes.

3.2.7 *Filtrado*

Esta es la última y más importante tarea en el proceso, ya que la localización nos deja con una media de 100000 a 200000 variantes. El objetivo es aislar unas pocas decenas de variantes, las cuales puedan ser sospechosas de causar la enfermedad.

El filtro más común es el filtro por frecuencias: aprovechando las anotaciones de SIFT y Polyphen-2, se eliminan aquellas variantes que tienen una frecuencia demasiado alta, superior a un 1 %, ya que se considera que con esta frecuencia la enfermedad afectaría a demasiadas personas. Además, se eliminan aquellas variantes que hayan sido leídas menos de 10 veces, que tengan valores de calidad por debajo de 50 y que no sean la alternativa más clara –si existen varias variantes en la misma posición–.

3.2.8 *Herramientas complementarias*

Estas herramientas no están asociadas a una etapa particular del procesamiento computacional de datos genómicos, pero son de gran utilidad para manejar ciertos tipos de ficheros.

Picard/samtools

SAM Tools[9] y Picard[20] son dos paquetes hermanos (SAMTools implementado en C; y Picard, en Java) que se utilizan para manipu-

lar ficheros en formato SAM y su binario BAM. Entre otras tareas, se puede ordenar, unir, indexar y generar alineamientos en otros formatos.

En muchos casos, aunque los localizadores de variantes admiten ficheros SAM/BAM, necesitan que estén previamente formateados. Esto no lo hacen los alineadores y para ello son muy útiles estos paquetes.

VCFTools

VCFTools[24] es un paquete de programas diseñado para trabajar con ficheros VCF, como los generados en el proyecto 1000 Genomas. El objetivo de VCFTools es suministrar métodos para validar, unir, comparar y calcular algunas estadísticas básicas en ficheros VCF.

3.3 SÍNTESIS DEL FLUJO DE TRABAJO

Tras evaluar detenidamente gran parte de los programas explicados en la sección anterior, se ha decidido seguir las recomendaciones del Broad Institute, ya que establece un flujo muy cercano a la estandarización, usando los formatos y algoritmos más comunes. Además, la mayor parte del software que se utiliza ha sido desarrollado por las mismas personas o por personas que trabajan en la misma organización. El flujo descrito en la Figura 3.2 es una adaptación del tutorial de buenas prácticas de GATK.[3]

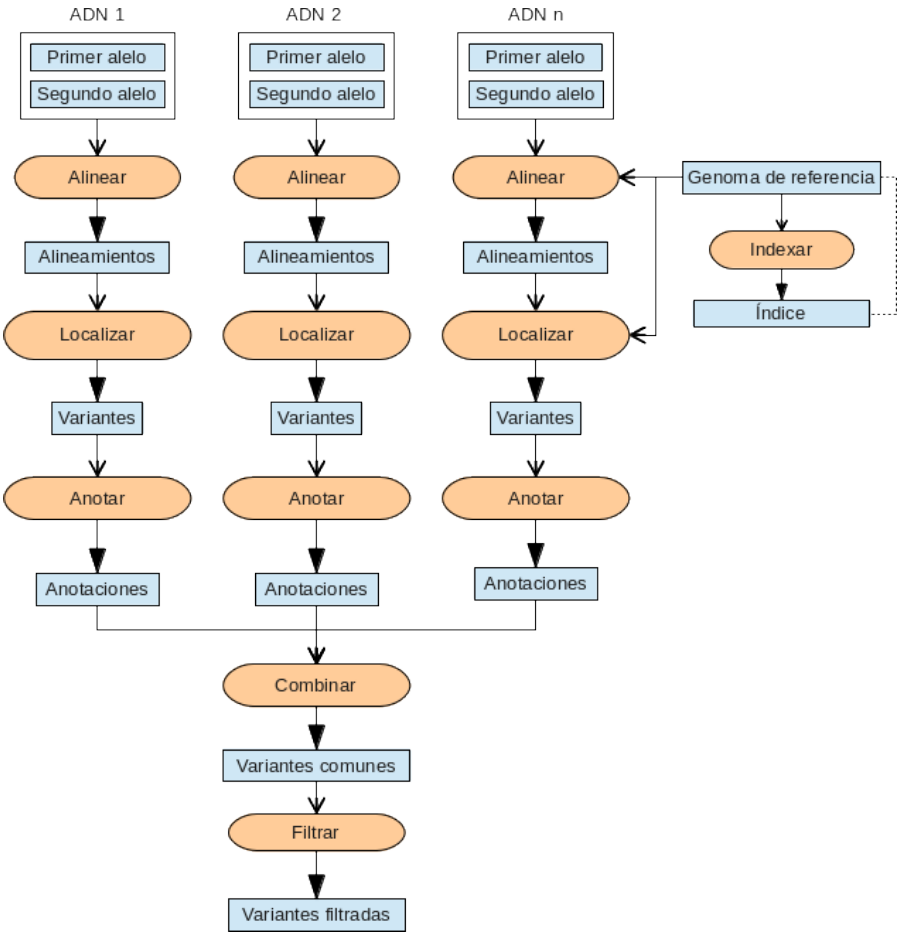


Figura 3.2.: *Flujo principal de trabajo*

3 http://gatkforums.broadinstitute.org/discussion/1186/best-practice-variant-detection-with-the-gatk-v4-for-release-2-0

3.3.1 *Indexar*

Antes de alinear cualquier secuencia, es necesario generar el índice del genoma de referencia. Mantener el índice en disco es un requisito de casi todos los programas alineadores.

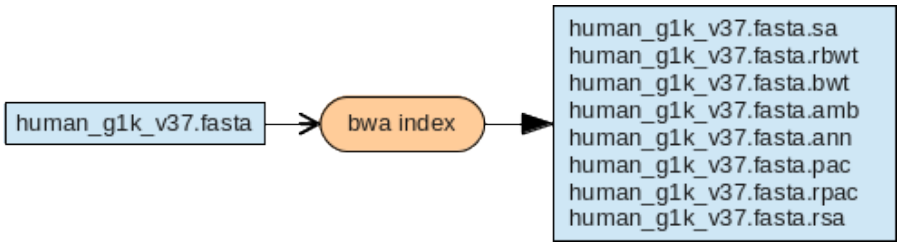


Figura 3.3.: *Indexación del genoma de referencia*

Nombre
bwa index
Entrada
referencia.fasta
Salida
referencia.amb, referencia.ann, referencia.bwt,referencia.pac, referencia.rbwt, referencia.rpac, referencia.rsa, referencia.sa
Descripción
El parámetro -a indica el algoritmo a utilizar, en este caso bwtsw se usa para genomas completos.
Comando
`bwa index –a bwtsw referencia.fasta`

3.3.2 *Alinear*

Por cada par de secuencias, hay que repetir el proceso de alineamiento, para obtener el genoma de la muestra. Aunque el alineamiento solo consiste en la primera parte del flujo mostrado en la Figura 3.4, es necesario ejecutar las siguientes operaciones para obtener un fichero normalizado y compatible con las fases posteriores.

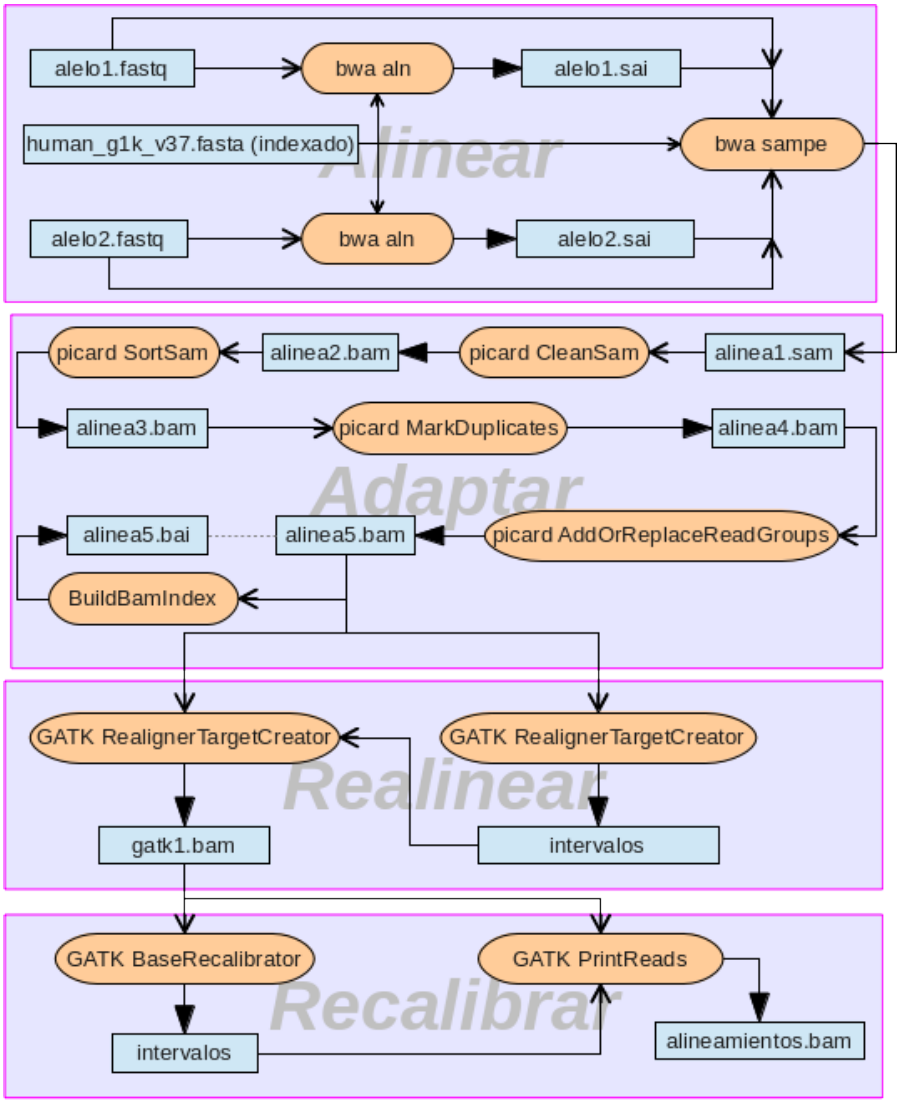


Figura 3.4.: *Alineamiento de secuencias*

Nombre
bwa aln
Entrada
secuencias.fastq
Salida
indices.sai
Descripción
Genera una lista con la posición de cada secuencia en el genoma de referencia. Debe hacerse con cada fichero de la pareja.
Comando
`bwa aln referencia.fasta secuencias1.fastq > indices1.sai`

Nombre
bwa sampe
Entrada
referencia.fasta, secuencias1.fastq, secuencias2.fastq, indices1.sai, indices2.sai
Salida
alineamientos.sam
Descripción
Genera un fichero que indica en cada posición del genoma de referencia qué secuencias están alineadas entre sí, teniendo en cuenta que ambos ficheros de entrada representan las dos cadenas del mismo ADN.
Comando
`bwa sampe referencia.fasta indices1.sai indices2.sai secuencias1.fastq secuencias2.fastq > alineamientos.sam`

Nombre
picard CleanSam
Entrada
alineamientos.sam
Salida
alineamientos1.bam
Descripción
CleanSam realiza dos tareas: poda los alineamientos que se salen del genoma de referencia y pone a cero las calidades de las secuencias no alineadas.
Comando

```
java -jar /ruta/hasta/CleanSam.jar \
   INPUT=alineamientos.sam \
   OUTPUT=alineamientos1.bam
```

Nombre
picard SortSam
Entrada
alineamientos1.bam
Salida
alineamientos2.bam
Descripción
Ordena los alineamientos según su posición en el genoma de referencia, de menor a mayor.
Comando

```
java -jar /ruta/hasta/SortSam.jar \
   INPUT=alineamientos1.bam \
   OUTPUT=alineamientos2.bam \
   SORT_ORDER=coordinate
```

Nombre
picard MarkDuplicates
Entrada
alineamientos2.bam
Salida
alineamientos3.bam
Descripción
Localiza y elimina alineamientos repetidos. Para cada serie de alineamientos repetidos, deja solo uno con la suma de todos los pesos.
Comando

```
java -jar /ruta/hasta/MarkDuplicates.jar \
  INPUT=alineamientos2.bam \
  OUTPUT=alineamientos3.bam \
  REMOVE_DUPLICATES=true \
  METRICS_FILE=metrics
```

Nombre
picard AddOrReplaceReadGroups
Entrada
alineamientos3.bam
Salida
alineamientos4.bam
Descripición
Repara o añade cabeceras en las secuencias.
Comando

```
java -jar /ruta/hasta/AddOrReplaceReadGroups.jar
    \
  INPUT=alineamientos3.bam \
  OUTPUT=alineamientos4.bam \
  RGPL=Illumina \
  RGPU=flowcellbarcode.lane \
  RGLB=BAITS \
  RGSM=nombre_muestra
```

Nombre
picard BuildBamIndex
Entrada
alineamientos4.bam
Salida
alineamientos4.sai
Descripción
Genera un índice del fichero BAM.
Comando

```
java -jar /ruta/hasta/BuildBamIndex.jar \
INPUT=alineamientos4.bam
```

Nombre
GATK RealignerTargetCreator
Entrada
referencia.fasta, alineamientos4.bam, indels.vcf, indels2.vcf
Salida
alineamientos5.bam
Descripción
Realinea los alineamientos para evitar falsos positivos. Las bases de datos conocidas suelen encontrarse en páginas de proyectos conocidos, como 1000 Genomes.
Comando

```
java -jar /ruta/hasta/GenomeAnalysisTK.jar \
    -T RealignerTargetCreator \
    -R referencia.fasta \
    -I alineamientos4.bam \
    -known indels.vcf \
    -known indels2.vcf \
    --fix_misencoded_quality_scores \
    -o intervals
java -jar /ruta/hasta/GenomeAnalysisTK.jar \
    -T IndelRealigner \
    -R referencia.fasta \
    -I alineamientos4.bam \
    -known indels.vcf \
    -known indels2.vcf \
    --fix_misencoded_quality_scores \
    -targetIntervals intervals \
    -o alineamientos5.bam
```

Nombre
GATK BaseRecalibrator
Entrada
referencia.fasta, alineamientos5.bam, snps.vcf, snps2.vcf
Salida
alineamientos5.bam
Descripción
Utilizando uno o varios ficheros que contienen snps conocidas, modifica las calidades de las secuencias basándose en un modelo probabilístico. Los ficheros con snps se utilizan para marcar zonas conocidas, ya que estos comandos trabajan solo en zonas desconocidas.
Comando

```
java -jar /ruta/hasta/GenomeAnalysisTK.jar \
    -T BaseRecalibrator \
    -R referencia.fasta \
    -I alineamientos5.bam \
    --knownSites snps.vcf \
    --knownSites snps2.vcf \
    -o recal
java -jar /ruta/hasta/GenomeAnalysisTK.jar \
    -T PrintReads \
    -R referencia.fasta \
    -I alineamientos5.bam \
    -BQSR recal \
    -o alineamientos6.bam
```

3.3.3 *Localizar*

Esta es la fase central del proceso. Una aproximación bastante fiel del exoma del paciente se encuentra en un fichero BAM. Ahora, es necesario volver a comparar esta cadena con el genoma de referencia para encontrar todas las variantes.

Nombre
GATK HaplotypeCaller
Entrada
referencia.fasta, alineamientos.bam, dbsnp.vcf
Salida
alineamientos5.bam
Descripción
Localiza simultáneamente SNPs e indels a través de un ensamblado de-novo de haplotipos. El fichero dbsnp.vcf se utiliza para etiquetar todas las variantes conocidas con un rsID, un código único para cada variante.
Comando

```
java -jar /ruta/hasta/GenomeAnalysisTK.jar \
    -T HaplotypeCaller \
    -R referencia.fasta \
    -I alineamientos.bam \
    --dbsnp dbsnp.vcf \
    -o salida.vcf
```

3.3.4 *Combinar*

Una vez obtenidas las variantes de todas las muestras, es el momento de utilizar las hipótesis establecidas. Se interseccionan, suman y restan variantes de cada fichero, hasta obtener uno más pequeño con aquellas variantes candidatas.

Nombre
GATK CombineVariants
Entrada
referencia.fasta, alineamientos.bam, dbsnp.vcf
Salida
alineamientos5.bam
Descripción
Localiza simultáneamente SNPs e indels a través de un ensamblado de-novo de haplotipos. El fichero dbsnp.vcf se utiliza para etiquetar todas las variantes conocidas con un rsID, un código único para cada variante.
Comando

```
//Interseccion
java -jar ruta/hasta/GenomeAnalysisTK.jar \
   -T CombineVariants \
   -R referencia.fasta \
   -minN numero_de_ficheros \
   -V variantes.vcf \
   -V variantes2.vcf \
   -o interseccion.vcf
// Suma
java -jar ruta/hasta/GenomeAnalysisTK.jar \
   -T CombineVariants \
   -R referencia.fasta \
   -V variantes.vcf \
   -V variantes2.vcf \
   -o interseccion.vcf
// Diferencia
java -jar ruta/hasta/GenomeAnalysisTK.jar \
   -T SelectVariants \
   -R referencia.fasta \
   -V variantes.vcf \
   --discordance variantes2.vcf \
   -o interseccion.vcf
```

3.3.5 *Anotar*

La fase de anotación se realiza online. Los ficheros con las variantes han de subirse a los distintos servidores de los programas a través de formularios personalizados. Los resultados suelen ser ficheros de texto separados por tabulador.

3.3.6 *Filtrar*

En este momento se aplican las hipótesis generadas al principio del proceso. Se calcula la intersección o la diferencia entre muestras, dejando solo las variantes de interés.

El filtrado se realiza a través de hojas de cálculo. Gran parte de las variantes se pueden elminiar ordenando las columnas de frecuencia y eliminando las que superan un umbral –1 por mil o 10 mil–. También se pueden eliminar las variantes sinónimas –no implican cambios en la proteína– y las toleradas –se sabe que no provocan enfermedades–.

A partir de ahí, empieza un proceso manual identificando los genes para conocer qué regiones del exoma codifican proteínas relevantes a la enfermedad de estudio.

Parte II

DESARROLLO DE LA APLICACIÓN

Donde se describe todo el proceso para la creación del programa informático DNAnalytics, desde su planificación hasta su implantación.

4

PLANIFICACIÓN

Este capítulo aborda el inicio del Plan de Sistemas. La planificación incluye la metodolgía de trabajo, la asignación de recursos y el calendario de tareas.

4.1 METODOLOGÍA DE DESARROLLO

Para desarrollar el software DNAnalytics se empleará una metodología de desarrollo ágil. Estas metodologías descomponen las tareas en pequeños incrementos de planificación mínima, donde cada iteración culmina con una entrega del software, aunque esté incompleta. Esto significa que todas son modelos iterativos e incrementales.

De este modo, desde muy temprano se puede liberar una versión del software funcional, aunque no completa, permitiendo al cliente probar y usar lo que ya se ha implementado. Así, se pueden cambiar, añadir o quitar requisitos desde el principio. Para los desarrolladores es muy fácil añadir pequeños cambios al software, y no hay que esperar mucho tiempo entre un lanzamiento y el siguiente.

Estas metodologías son ideales para proyectos que no tengan los requisitos claros desde un principio, y que puedan usarse aunque solo funcione parte del sistema.

En el año 2001 se creó el *Manifiesto por el desarrollo ágil* con el objeto de mejorar los resultados en el desarrollo del software, no solo en reducción de tiempo, sino también en número de proyectos exitosos.

Los cuatro pilares del manifiesto ágil son los siguientes:

1. Individuos e interacciones sobre procesos y herramientas
2. Software funcionando sobre documentación extensiva
3. Colaboración con el cliente sobre negociación contractual
4. Respuesta ante el cambio sobre seguir un plan

Y sus doce principios son los que siguen:

1. Nuestra mayor prioridad es satisfacer al cliente con tempranas y continuas entregas de software valioso.

2. Los cambios en los requisitos son bienvenidos, incluso tarde en el desarrollo. Los procesos ágiles aprovechan los cambios para proporcionar ventajas competitivas al cliente.

3. Entregamos software funcional con frecuencia, entre dos semanas y dos meses, preferentemente la menor escala de tiempo.

4. Administradores y desarrolladores deben trabajar juntos día a día durante el proyecto.

5. Construir proyectos entre individuos motivados. Darles el entorno y el apoyo que necesitan, y confiar en que hagan el trabajo.

6. El método más eficiente y efectivo para comunicar información hacia el equipo de desarrollo y entre sus miembros es la conversación cara a cara.

7. La mejor medida de progreso es el software funcional.

8. Los procesos ágiles promueven el desarrollo sostenible. Promotores, desarrolladores y usarios deben ser capaces de mantener un ritmo constante indefinidamente.

9. La atención continua a la excelencia técnica y al buen diseño mejora la agilidad.

10. La simplicidad, el arte de maximizar la cantidad de trabajo no hecho es esencial.

11. Las mejores arquitecturas, requisitos y diseños surgen de equipos auto-organizados.

12. A intervalos regulares, el equipo reflexiona cómo puede ser más efectivo, entonces afina y ajusta su comportamiento en consecuencia.

La principal desventaja de estas metodologías es que las versiones tienen una fecha de caducidad muy corta, y obligan al cliente a estar actualizando o cambiando el sistema constantemente. Por ello, los desarrolladores deben tener claro que un nuevo lanzamiento supone una mejora crítica o necesaria, para así mantener el interés del cliente.

La elección de usar una metodología ágil no es arbitraria. Como no existe un estándar en el tratamiento computacional del ADN, es preferible un marco que permita lanzar prototipos de manera rápida para ser sometidos a pruebas lo antes posible. De este modo, se podrán satisfacer los requisitos del usuario a medida que estos aparezcan.

En este proyecto, al no existir más que un desarrollador y un cliente, no se aplicará estrictamente ninguna metodología existente, pero se intentará simular las características principales:

- *Reuniones periódicas:* Intentar reunirse a diario o, al menos, semanalmente. Estas reuniones servirán para poner en común los progresos y establecer el trabajo próximo.
- *Lanzamientos frecuentes:* Lanzar una versión de la aplicación al menos una vez al mes.
- *Incrementos progresivos:* Cada versión debe suponer una mejora significativa respecto a la anterior.
- *Modificación ágil de requisitos:* Asimilación rápida y positiva de los cambios en los requisitos.

4.2 PLATAFORMAS DE DESARROLLO

Para la etapa de diseño se utilizará el lenguaje de modelado Unified Modeling Language (UML, Lenguaje de Modelado Unificado), desarrollado por el Object Management Group (OMG), un consorcio sin ánimo de lucro para el desarrollo de estándares en la industria informática. UML ayuda a especificar, visualizar y documentar modelos de sistemas informáticos, incluyendo su estructura y diseño, de acuerdo a los requisitos. El modelado se realiza a través de hasta 13 tipos de diagramas diferentes: 6 para representar la estructura estática de la aplicación, 3 para el comportamiento general y 4 para interacciones.

El código de programación se escribirá principalmente en Java, empleando la plataforma JavaFX. Esta plataforma, diseñada por Oracle[2], permite generar interfaces de usuario empleando un lenguaje de etiquetado, FXML, que es compatible con XML, permitiendo separar completamente la vista del modelo. FXML se integra de forma natural con Cascade Style Sheet (CSS), de este modo, la vista permite un alto grado de modificación sin alterar el resto del sistema. Además, facilita la exportación entre software de escritorio y software web.

4.3 RECURSOS

A continuación se detallan los recursos materiales, tanto hardware como software, que se utilizan en el desarrollo del programa informático DNAnalytics.

4.3.1 *Recursos hardware*

El alto coste computacional del alineamiento del ADN requiere de mucha potencia de procesamiento y mucha capacidad de memoria. El software analizado y seleccionado requiere de aproximadamente 8 a 10 GB de memoria. En cuanto al procesador, debe ser lo más potente posible, y multinúcleo, para analizar datos en paralelo. Es necesaria, al menos, una máquina con estas características, mientras que se puede utilizar una unidad de almacenamiento secundaria para conservar los datos.

En la Tabla 4.1 se muestran las características de la máquina propuesta:

Elemento	Descripción
Memoria RAM	16 ó 32 GB DDR3-1333/1600
CPU	4 o más núcleos a 3 o más GHz, 64 bits
Almacenamiento	128 GB SSD + 1 TB HDD

Tabla 4.1.: *Recursos hardware*

4.3.2 *Recursos software*

Gran parte de los programas desarrollados para el análisis genético está desarrollado en un entorno Linux, de modo que se opta por un sistema operativo Linux, en este caso la distribución CentOS Release 6.4, por ser una distribución orientada al desarrollo y la productividad.

Como el lenguaje de programación elegido ha sido JavaFX, y por recomendación de la página web oficial de JavaFX, se utilizará el entorno de programación NetBeans IDE 7, así como la máquina virtual de Java.

Para la vista se usa el JavaFx Scene Builder, un programa que genera interfaces gráficas de usuario con el sistema arrastrar y soltar.

El código FXML que genera puede ser utilizado por JavaFX, de modo que durante el diseño ya se genera la mayor parte de la vista del programa.

Además, ya que DNAnalytics es un programa orientado a la creación de flujos de programa (scripting), se instalarán en el equipo los programas de los que depende: Burrows Wheeler Aligner (BWA), samtools/picard, The Genome Analysis Toolkit (GATK), perl y python.

4.4 PLAN DE TRABAJO

El proyecto se divide en 3 fases principales en las que se desarrollan distintas tareas. A continuación se ofrece una explicación de cada una de ellas, con su estimación temporal:

4.4.1 *Fase 1: análisis*

Preparación del proyecto de desarrollo del sistema de información. Aquí se estudia la magnitud del proyecto, se estiman los recursos y se genera el plan de trabajo. El producto entregable es este mismo capítulo: Capítulo 4. La estimación temporal se encuentra en la Tabla 4.2.

Elaboración del plan de trabajo	25 horas
Asignación de recursos	75 horas
Catalogación inicial de requisitos	150 horas
Total	250 horas

Tabla 4.2.: *Planificación de la fase 1: análisis*

4.4.2 *Fase 2: desarrollo*

Implementación del sistema de información. Al ser un proceso iterativo y evolutivo, se presenta en varias iteraciones, en cada una de las cuales se implementan los requisitos que aún no se cumplen. El desarrollo culmina cuando todos los requisitos han sido validados. Cada etapa, a su vez, se divide en tres secciones: implementación, pruebas y modificación de requisitos. El producto entregable es el Capítulo 5. La estimación temporal se encuentra en la Tabla 4.3.

Diseño inicial.	80 horas
Iteración 1: Requisitos vitales.	120 horas.
Iteración 2: Requisitos de menor prioridad.	90 horas
Iteración 3: Modificación de requisitos.	120 horas
Iteración 4: Diseño final	90 horas
Total	500 horas

Tabla 4.3.: *Planificación de la fase 2: desarrollo*

4.4.3 *Fase 3: publicación*

Implantación y aceptación del sistema de información. Ya que en cada fase del desarrollo se han ejecutado las pruebas, no es necesario volver a realizarlas. El capítulo 7 muestra un ejemplo del uso del programa DNAnalytics. En la Tabla 4.4 se muestra la estimación temporal de esta fase.

Validación	50 horas
Instalación	10 horas
Manual de usuario	25 horas
Mantenimiento evolutivo	75 horas
Total	160 horas

Tabla 4.4.: *Planificación de la fase 3: publicación*

4.5 COSTES DEL PROYECTO

Los costes de este proyecto se miden en dos partes fácilmente diferenciables: el coste humano, donde se incluyen los salarios y costes de manutención y transporte; y el coste material, donde se incluyen los recursos hardware, el material de oficina y otros consumibles.

4.5.1 *Recursos humanos*

Para simplificar los cálculos se incluyen los gastos de transporte y manutención en el salario, y se establece el gasto por hora de 25 € para un programador/diseñador que trabaja en todas las fases del proyecto y de 40 € para un jefe de investigación que trabaja en la fase de análisis y en la validación del programa. Si estimamos los

gastos de seguridad social e impuestos como un incremento del 50 por ciento, el gasto total será de 48750 €, detallado en la Tabla 4.5.

Programador	900 horas * 25€/hora * 1,5	33750 €
Jefe de investigación	250 horas * 40€/hora * 1,5	15000€
	Total	48750€

Tabla 4.5.: *Coste de personal humano*

4.5.2 *Material*

En la Tabla 4.6 se incluyen los gastos de oficina y los recursos hardware necesarios.

Material fungible	
Papelería (papel, tinta, encuadernación...)	150€
Material inventariable	
Computadora	2000 €
Disco duro externo	150 €
TOTAL	2300€

Tabla 4.6.: *Coste de material*

Lo que hace un coste total del proyecto de 51050 €.

5

DESARROLLO

En este capítulo se explica el proceso de creación de la aplicación DNAnalytics. El desarrollo consiste en la ejecución de lo planificado, procurando mantener los tiempos marcados y utilizando los recursos asignados de manera adecuada. El producto final de esta fase es la aplicación software y la catalogación formal de los objetivos y requisitos del sistema.

5.1 LISTA DE CAMBIOS

La lista de cambios –Tabla 5.1– contiene las versiones del programa ordenadas cronológicamente, explicando los cambios realizados en cada una de ellas. Para cada versión se explican, primero, los cambios y, finalmente, el número de versión.

Nº	Fecha	Autor/es	Descripción
00	30/11/2012	P. Lorente A. Tugores	Se crean las herramientas: alinear ADN y localizar variantes.
01	30/11/2012	P. Lorente	Versión 0.1.
02	15/01/2013	P. Lorente	La interfaz gráfica se genera ahora con FXML.
03	15/01/2013	P. Lorente	Las bases de datos se seleccionan en cada herramienta, y no en la pestaña de configuración.
04	15/01/2013	P. Lorente A. Tugores	Añadidas herramientas: seleccionar variantes, anotar variantes, recodificar secuencias.
05	15/01/2013	P. Lorente	La barra de progreso muestra progreso, ya no es indeterminada.
06	15/01/2013	P. Lorente	Versión 0.2.
07	25/02/2013	P. Lorente A. Tugores	Nuevas herramientas: combinar variantes, filtrar variantes.
08	25/02/2013	P. Lorente	Se elimina la herramienta recodificar secuencias.
09	25/02/2013	P. Lorente	Se añade el idioma español.
10	25/02/2013	P. Lorente	Se añade la selección de carpeta temporal.
11	25/02/2013	P. Lorente	Versión 0.5.
12	20/04/2013	P. Lorente	Multitarea con pestañas individuales para cada proceso.
13	20/04/2013	P. Lorente	Alinear secuencias y localizar variantes son degradadas a herramientas.
14	20/04/2013	P. Lorente	Versión 1.0.

Tabla 5.1.: *Lista de cambios*

5.2 OBJETIVOS

Los objetivos representan las necesidades principales del cliente y describen a alto nivel las características básicas del sistema. Los objetivos catalogados deben ser desarrollados y validados a través de los requisitos. El proceso de desarrollo se considera finalizado cuando todos los objetivos han sido validados.

OBJ–01	Localización de variantes
Versión	1.0 (20/04/2013)
Autores	Pascual Lorente Arencibia Antonio Tugores Cester
Descripción	El sistema debe permitir buscar variantes en las secuencias de ADN de un paciente.
Importancia	vital
Estado	Validado
Urgencia	Alta
Estabilidad	Alta

Tabla 5.2.: *Objetivo OBJ-01*

OBJ–02	Alineado de secuencias
Versión	1.0 (20/04/2013)
Autores	Pascual Lorente Arencibia Antonio Tugores Cester
Descripción	El sistema debe alinear los ficheros de secuencias de ADN de una muestra.
Importancia	vital
Estado	Validado
Urgencia	Alta
Estabilidad	Alta

Tabla 5.3.: *Objetivo OBJ-02*

OBJ–03	Filtrado de variantes
Versión	1.0 (20/04/2013)
Autores	Pascual Lorente Arencibia Antonio Tugores Cester
Descripción	El sistema debe permitir filtrar un grupo de variantes en base a su frecuencia o tipo.
Importancia	vital
Estado	Validado
Urgencia	Media
Estabilidad	Alta

Tabla 5.4.: *Objetivo OBJ-03*

5.3 REQUISITOS

Los requisitos definen formalmente el funcionamiento del proyecto. Cada requisito representa una característica o funcionalidad que debe

poseer el programa. Existen varios tipos de requisitos para ajustarse a las diversas características de las aplicaciones software.

5.3.1 *Requisitos funcionales*

Definen las funcionalidades o capacidades que debe ofrecer el sistema a los usuarios para alcanzar los objetivos.

RF-01	Mostrar progreso
Versión	1.0 (20/04/2013)
Autores	Pascual Lorente Arencibia
Descripción	El sistema deberá mostrar al usuario el progreso de cada ejecución a través de una barra de progreso y una barra de estado.
Estado	Validado
Estabilidad	Alta

Tabla 5.5.: *Requisito funcional RF-01*

RF-02	Guardar configuración
Versión	1.0 (20/04/2013)
Autores	Pascual Lorente Arencibia
Descripción	El sistema deberá guardar automáticamente en disco: rutas de las bases de datos, del genoma de referencia, de la carpeta temporal e idioma.
Estado	Validado
Estabilidad	Alta

Tabla 5.6.: *Requisito funcional RF-02*

5.3.2 *Casos de uso*

Los casos de uso son un tipo de requisito funcional. Se utilizan para representar las tareas que realiza cada actor y cómo interactúan con el sistema. En este caso, el único actor del sistema podrá realizar todas las tareas.

ACT–01	Usuario
Versión	0.1 (30/10/2012)
Autores	Pascual Lorente Arencibia
Descripción	Este actor representa al único rol del programa.

Tabla 5.7.: *Caso de uso CU-01*

La figura 5.1 muestra un resumen de todos los casos de uso que puede realizar el usuario utlizando un esquema UML. Solo se representan aquellos casos de uso que permanecen en el programa final.

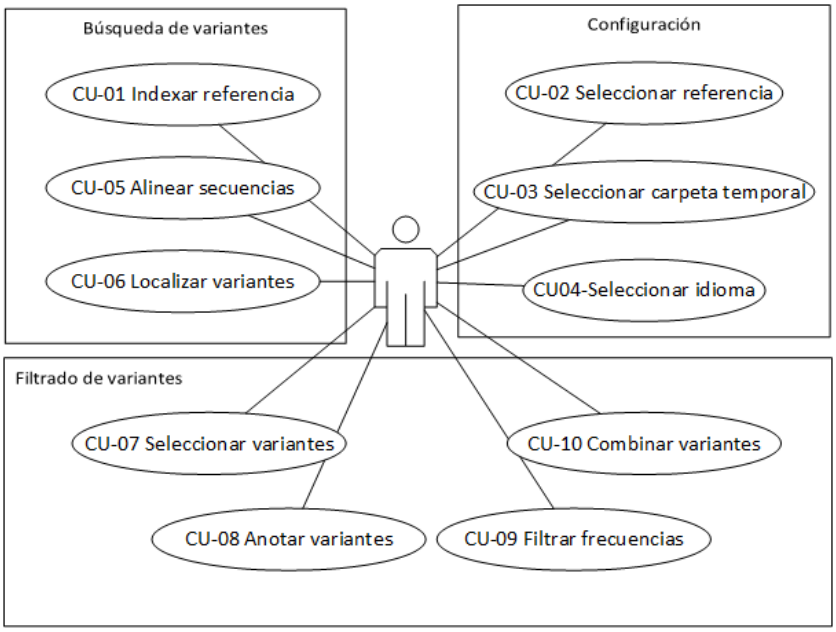


Figura 5.1.: *Casos de uso versión 1.0 (20/04/2013)* .

A continuación se detallan los casos de uso, describiendo el flujo de acciones que se debe realizar en cada uno de ellos. Para el CU-01 se ha hecho una captura de pantalla completa, para los demás, solo de la parte específica. Los casos de uso CU-02, CU-03 y CU-04 pueden verse en la misma captura.

CU-01	Indexar referencia
Versión	1.0 (20/04/2013)
Autores	Pascual Lorente Arencibia
Actor principal	Usuario
Descripción	El usuario indexa el genoma de referencia.
Poscondición	Se crean varios ficheros con el índice generados (.amb, .ann, .bwt, .pac, .rbwt, .rpac, .rsa, .sa)
Flujo normal	
1. El usuario hace clic sobre la pestaña «Herramientas». 2. El usuario selecciona la herramienta «Indexar genoma». 3. El usuario selecciona el genoma de referencia. 4. El usuario hace clic en empezar. 5. El sistema muestra una pestaña nueva en la consola con el progreso. [5]. El usuario hace clic sobre «Cancelar» para detener el proceso.	

Tabla 5.8.: *Caso de uso CU-01*

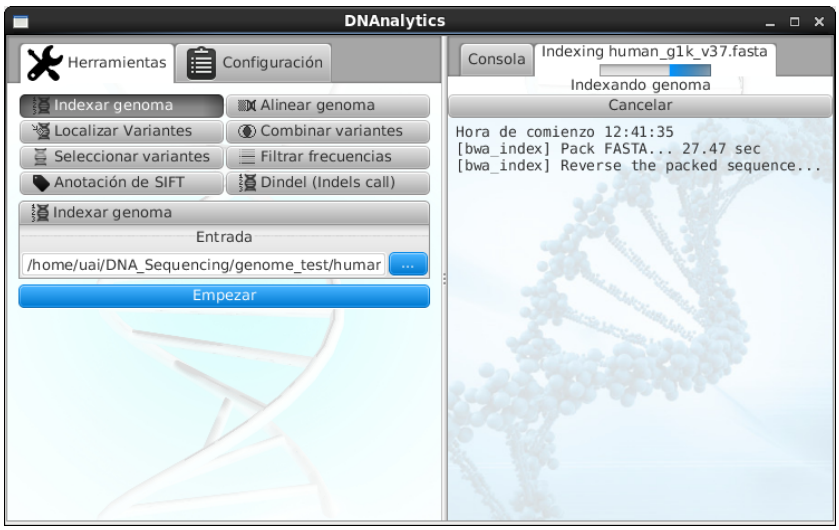


Figura 5.2.: *Vista del caso de uso CU-01.*

CU-02	Seleccionar genoma de referencia
Versión	1.0 (20/04/2013)
Autores	Pascual Lorente Arencibia
Actor principal	Usuario
Descripción	El usuario selecciona el genoma de referencia.
Flujo normal	
1. El usuario hace clic sobre la pestaña «Configuración». 2. El usuario selecciona el genoma de referencia.	

Tabla 5.9.: *Caso de uso CU-02*

CU-03	Seleccionar carpeta temporal
Versión	1.0 (20/04/2013)
Autores	Pascual Lorente Arencibia
Actor principal	Usuario
Descripción	El usuario selecciona la carpeta temporal.
Flujo normal	
1. El usuario hace clic sobre la pestaña «Configuración». 2. El usuario selecciona la carpeta temporal.	

Tabla 5.10.: *Caso de uso CU-03*

CU-04	Seleccionar idioma
Versión	1.0 (20/04/2013)
Autores	Pascual Lorente Arencibia
Actor principal	Usuario
Descripción	El usuario selecciona el idioma del sistema.
Flujo normal	
1. El usuario hace clic sobre la pestaña «Configuración». 2. El usuario selecciona el idioma deseado. 3. El usuario reinicia la aplicación. 4. El sistema se muestra en el último idioma seleccionado.	

Tabla 5.11.: *Caso de uso CU-04*

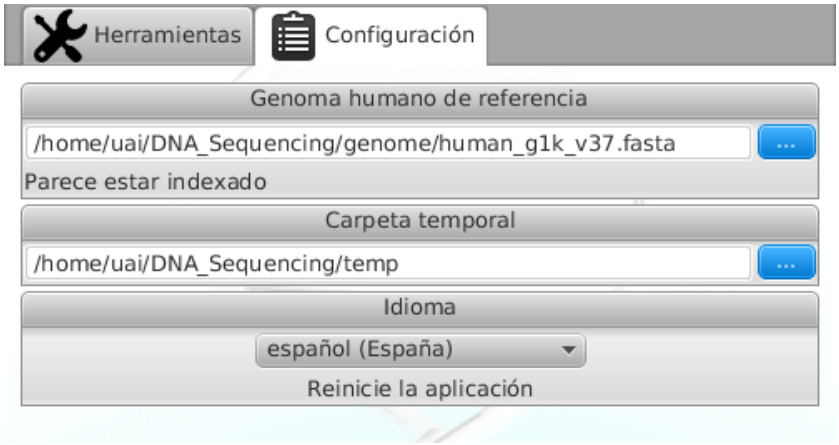


Figura 5.3.: *Vista de los casos de uso CU-02, CU-03 y CU-04.*

CU-05	Alinear ADN
Versión	1.0 (20/04/2013)
Autores	Pascual Lorente Arencibia
Actor principal	Usuario
Descripción	El usuario alinea dos ficheros .fastq.
Postcondición	Se crea un fichero .bam
Flujo normal	
1. El usuario selecciona la pestaña «Herramientas». 2. El usuario selecciona la herramienta «Alinear genoma». 3. El usuario selecciona los ficheros de entrada. 4. El usuario selecciona las bases de datos de entrenamiento. 5. El usuario selecciona el fichero de salida. 6. El usario selecciona las opciones. 7. El usuario comienza el alineado haciendo clic en «Empezar». 7. El sistema crea una pestaña con el progreso. [8]. El usuario cancela el proceso haciendo clic en «Cancelar».	

Tabla 5.12.: *Caso de uso CU-05*

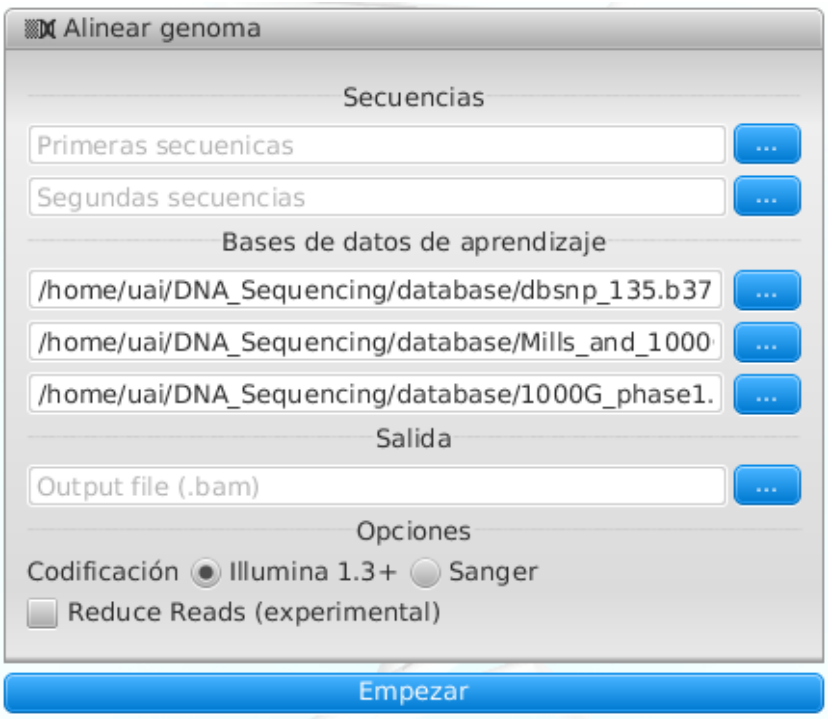


Figura 5.4.: *Vista del caso de uso CU-05.*

CU-06	Localizar variantes
Versión	1.0 (20/04/2013)
Autores	Pascual Lorente Arencibia
Actor principal	Usuario
Descripción	El usuario busca las variantes en un fichero .bam.
Postcondición	Se crea un fichero .vcf
Flujo normal	
1. El usuario selecciona la pestaña «Herramientas». 2. El usuario selecciona la herramienta «Localizar variantes». 3. El usuario selecciona el fichero de entrada. 4. El usuario selecciona el fichero de salida. 5. El usuario elige si quiere recalibrar la salida. 5.1. El usuario selecciona los ficheros de recalibrado. 7. El usuario hace clic en «Empezar». 7. El sistema crea una pestaña con el progreso. [8]. El usuario cancela el proceso haciendo clic en «Cancelar».	

Tabla 5.13.: *Caso de uso CU-06*

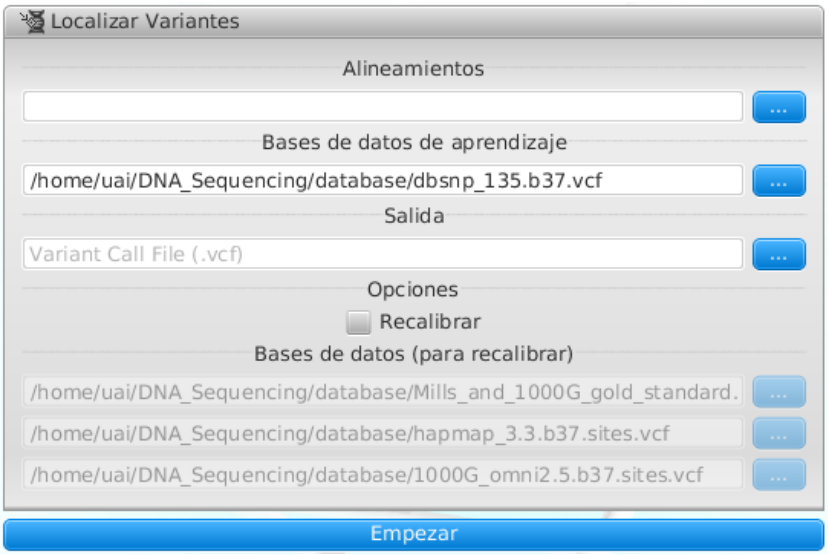


Figura 5.5.: *Vista del caso de uso CU-06.*

CU-07	Seleccionar variantes
Versión	1.0 (20/04/2013)
Autores	Pascual Lorente Arencibia
Actor principal	Usuario
Descripción	El usuario toma un fichero de variantes y las filtra en base a una expresión JEXL.
Flujo normal	
1. El usuario selecciona la pestaña «Herramientas». 2. El usuario selecciona la herramienta «Seleccionar variantes». 3. El usuario selecciona el fichero VCF de entrada. 4. El usuario selecciona el fichero de salida. 6. El usuario la expresión JEXL de filtrado. 7. El usuario hace clic en «Empezar». 8. El sistema muestra el progreso en la consola. [9]. El usuario cancela el proceso haciendo clic en «Cancelar».	

Tabla 5.14.: *Caso de uso CU-07*

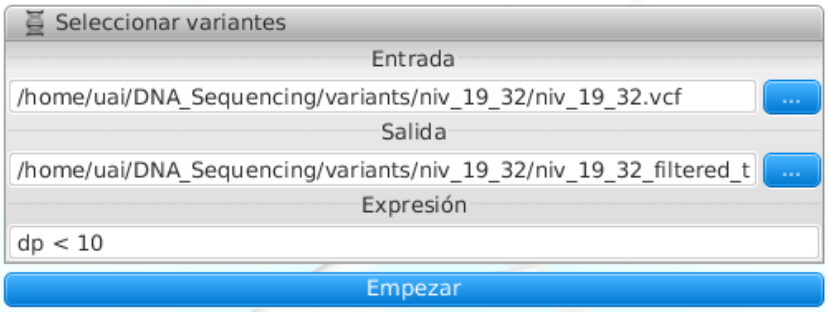


Figura 5.6.: *Vista del caso de uso CU-07.*

CU-08	Anotar variantes
Versión	1.0 (20/04/2013)
Autores	Pascual Lorente Arencibia
Actor principal	Usuario
Descripción	El usuario obtiene dos ficheros tabulados–uno de SNPs, otro de InDels– a partir de una lista de variantes. Estos ficheros contienen las anotaciones separadas por tabulador.
Flujo normal	
1. El usuario selecciona la pestaña «Herramientas». 2. El usuario selecciona la herramienta «Anotación SIFT». 3. El usuario selecciona cada fase de anotación y sigue las instrucciones de la página web donde es redirigido.	

Tabla 5.15.: *Caso de uso CU-08*

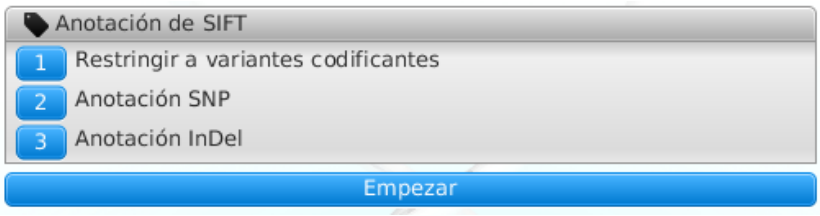


Figura 5.7.: *Vista del caso de uso CU-08.*

CU-09	Filtrar por frecuencias
Versión	1.0 (20/04/2013)
Autores	Pascual Lorente Arencibia
Actor principal	Usuario
Descripción	El usuario elimina de un fichero anotado las filas que superan un umbral de frecuencia en alguna de sus columnas.
Flujo normal	
1. El usuario selecciona la pestaña «Herramientas». 2. El usuario selecciona la herramienta «Filtrar frecuencias». 3. El usuario selecciona la columna a filtrar. 4. El usuario selecciona la frecuencia umbral. 5. El usuario hace clic en «Empezar». 6. El El sistema muestra el progreso en una consola nueva.	

Tabla 5.16.: *Caso de uso CU-09*

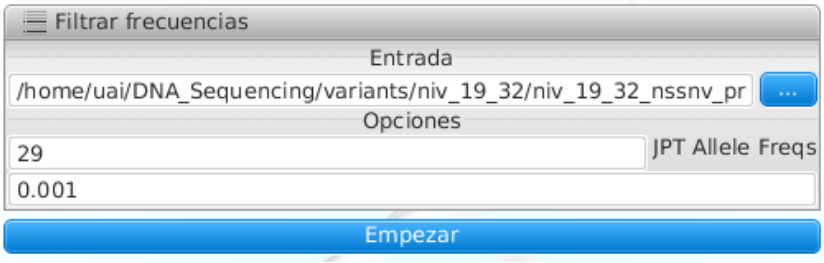


Figura 5.8.: *Vista del caso de uso CU-09.*

CU-10	Combinar variantes
Versión	1.0 (20/04/2013)
Autores	Pascual Lorente Arencibia
Actor principal	Usuario
Descripción	El usuario intersecciona, suma o diferencia las variantes de 2 o más pacientes.
Flujo normal	
1. El usuario selecciona la pestaña «Herramientas». 2. El usuario selecciona la herramienta «Combinar variantes». 3.1. El usuario añade ficheros con variantes con «Añadir». 3.2. El usuario elimina ficheros con variantes con «Eliminar». 4. El usuario selecciona el tipo de combinación. 5. El usuario hace clic en «Empezar». 6. El El sistema muestra el progreso en una consola nueva.	

Tabla 5.17.: *Caso de uso CU-10*

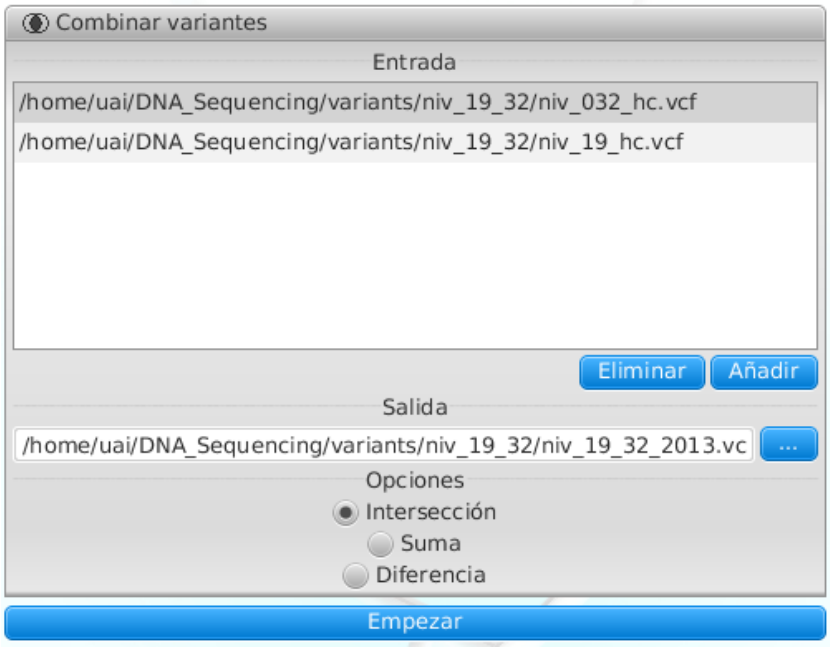


Figura 5.9.: *Vista del caso de uso CU-10.*

5.3.3 *Requisitos no funcionales*

Los requisitos no funcionales representan aquellas características relacionadas con la calidad, la fiabilidad y el rendimiento del sistema. Surgen de la necesidad del usuario, debido a las restricciones en el presupuesto, a las políticas de la organización, a la necesidad de interoperabilidad con otros sistemas de software o hardware o a fac-

tores externos como los reglamentos de seguridad o las políticas de privacidad, entre otros.

RNF-01	Multilenguaje
Versión	1.0 (20/04/2013)
Autores	Pascual Lorente Arencibia
Descripción	El sistema deberá estar traducido, al menos, a dos idiomas: español e inglés.
Estado	Validado
Estabilidad	Alta

Tabla 5.18.: *Requisito no funcional RNF-01*

RNF-02	Rendimiento mínimo.
Versión	1.0 (20/04/2013)
Autores	Pascual Lorente Arencibia
Descripción	La diferencia de tiempo entre la ejecución de las tareas en DNAnalytics y la ejecución de las mismas en una consola de comandos no debe superar el 5 % del tiempo total de ejecución en la consola de comandos. Es decir: $\frac{T_{DNAnalytics} - T_{consola}}{T_{consola}} < 0,05$
Estado	Validado
Estabilidad	Media

Tabla 5.19.: *Requisito no funcional RNF-02*

RNF-03	Memoria máxima.
Versión	1.0 (20/04/2013)
Autores	Pascual Lorente Arencibia
Descripción	El programa no puede utilizar nunca más de 8GB de memoria por tarea. Nunca se debe permitir el uso de memoria virtual.
Estado	Validado
Estabilidad	Media

Tabla 5.20.: *Requisito no funcional RNF-03*

5.4 MATRIZ DE RASTREABILIDAD

La matriz de rastreabilidad –Figura 5.21– indica qué requisitos resuelven cada uno de los objetivos.

	OBJ-01	OBJ-02	OBJ-03
RF-01	•	•	•
RF-02	•	•	
CU-01		•	
CU-02	•	•	
CU-03	•	•	
CU-04	•	•	•
CU-05		•	
CU-06	•		
CU-07			•
CU-08			•
CU-09			•
CU-10			•
RNF-01	•	•	•
RNF-02	•	•	•
RNF-03	•	•	•

Tabla 5.21.: *Matriz de rastreabilidad*

5.5 DIAGRAMA DE CLASES

Este diagrama –Figura 5.10– representa la implementación en código del programa. En él se dibujan las principales clases y las relaciones que hay entre ellas.

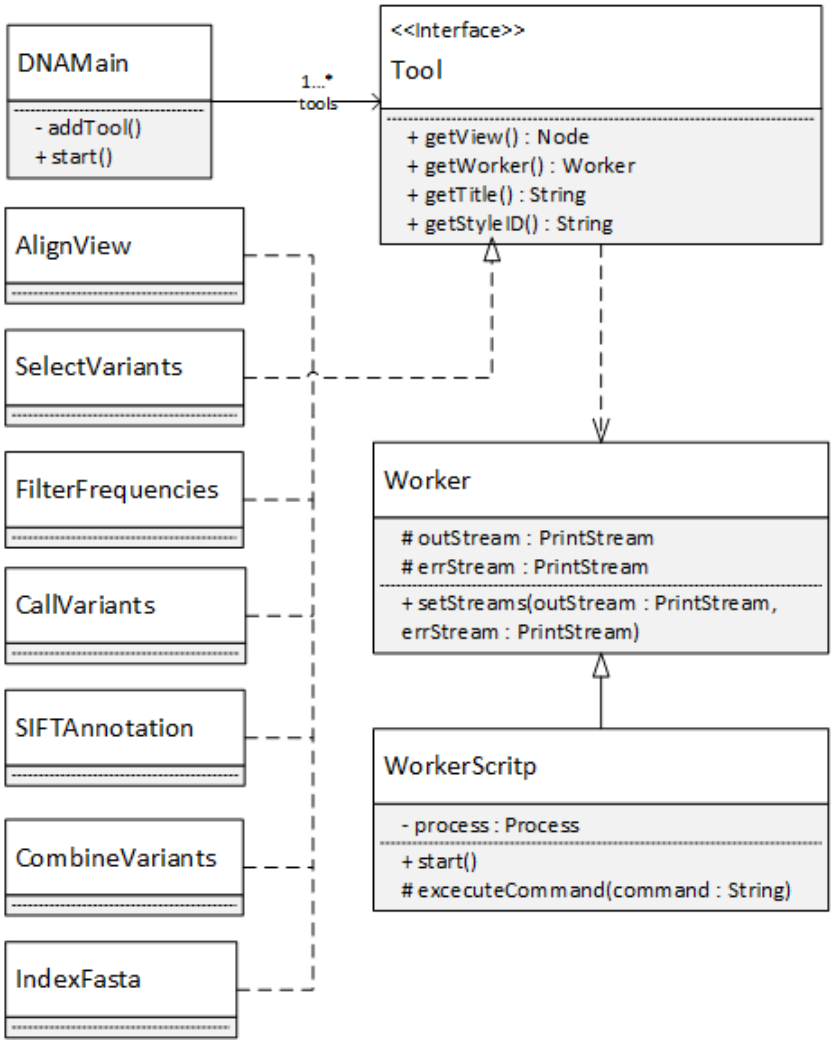


Figura 5.10.: *Diagrama de clases.*

- *DNAMain:* Es el controlador de la vista y contiene una lista de herramientas, de la clase *Tool*, añadidas con el método *addTool()*. A petición del usuario, lanzará el *Worker* de la herramienta seleccionada en la vista a través del método *start()*.

- *Tool:* Esta clase pretende empaquetar la vista, el controlador y el proceso de cada herramienta. La vista es un objeto de clase *Node* de las librerías gráficas de JavaFX, el proceso es un *Worker* y el controlador puede ser ella misma, o designado por la vista.

- *Worker:* Clase abstracta para ejecutar los procesos del usuario que extiende la clase *Task* de Java. Encapsula los búferes de salida de las subclases.

- *WorkerScript:* Extiende la clase *Worker* para que solo sea necesario implementar el método *start()*. A través del método *execu-*

teCommand(), se ejecutan de forma controlada comandos en el sistema operativo –scripting–.

De manera general, cada vez que se quiera añadir una herramienta a DNAnalytics, basta con implementar los métodos de la interfaz *Tool*, lo que obliga a crear tanto la vista como el proceso. La nueva herramienta se añade al vector de herramientas de *DNAMain*.

Parte III

RESULTADOS Y CONCLUSIONES

Donde se presenta el uso de DNAnalytics en un ejemplo,
y se expresan las conclusiones del trabajo.

6

CASO PRÁCTICO

Para concluir, en este capítulo se pretende mostrar un ejemplo de cómo se utiliza el programa DNAnalytics empleando un caso práctico.

En primer lugar, partimos del árbol genealógico de la familia afectada, cuyo código es NIV, en la Figura 6.1, para establecer las hipótesis.

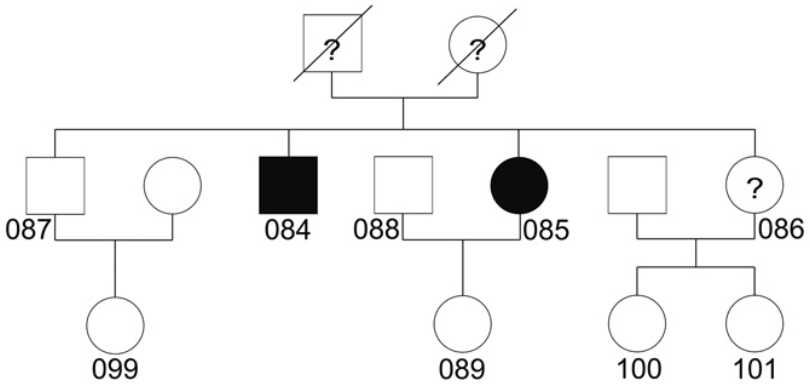


Figura 6.1.: *Árbol genealógico de NIV.*

- *Hipótesis 1: homocigota recesiva*. Los dos alelos presentan la variante. Exigiría que ambos progenitores portaran la enfermedad, pudiendo padecerla o no, aunque la frecuencia en los hermanos es anómala para ser mendeliana, 2 de 4 hermanos enfermos.

- *Hipótesis 2: heterocigota dominante*. Solo uno de los alelos presenta la variante. Eso explicaría que la mitad de los hermanos tuviera la enfermedad.

Para ello, se envían a secuenciar muestras de ADN de los pacientes 84 y 85, obteniendo 56.000.000 secuencias para 84 y 56.931.564 para 85, que ocupan, comprimidos, 2,1 GB de espacio en disco.

Las secuencias son alineadas con DNAnalytics –herramienta *Alinear ADN*–, obteniendo los ficheros BAM normalizados, de 7,8 GB para 84 y 6,7 GB para 85.

En este momento se ejecuta la localización de variantes –herramienta *Localizar variantes*–, obteniendo 149.849 variantes en 84 y 133.995 en 85.

Como la hipótesis más viable es la Hipótesis 2, se localizan las variantes heterocigotas en cada fichero. Con *Seleccionar variantes*, utlizando la expresión *AF==0,5*, se eliminan las no heterocigotas, quedando 71.019 variantes en 84 y 60.566 variantes en 85.

El siguiente paso es calcular las variantes comunes entre 84 y 85 con la intersección de *Combinar variantes*, lo que resulta en 26.612 variantes.

Ahora se procede a anotar con SIFT via web, la herramienta *Anotación de SIFT* nos enlaza ordenadamente a los distintos pasos en la web de SIFT. Como SIFT restringe las variantes a aquellas que se encuentran dentro de los genes, también se produce un filtrado. Al final, se obtienen 988 variantes InDel y 1583 SNPs.

Una vez tenemos las anotaciones en formato TSV, utilizamos *Filtrar frecuencias* y columna a columna, eliminamos las variantes con una frecuencia superior a 1 por mil, con un total de 195 SNPs y 405 InDels.

Por último, un biólogo ha de revisar una a una las variantes restantes, analizando si influyen en la enfermedad de estudio.

Un esquema orientativo de este proceso puede verse en la imagen 6.2.

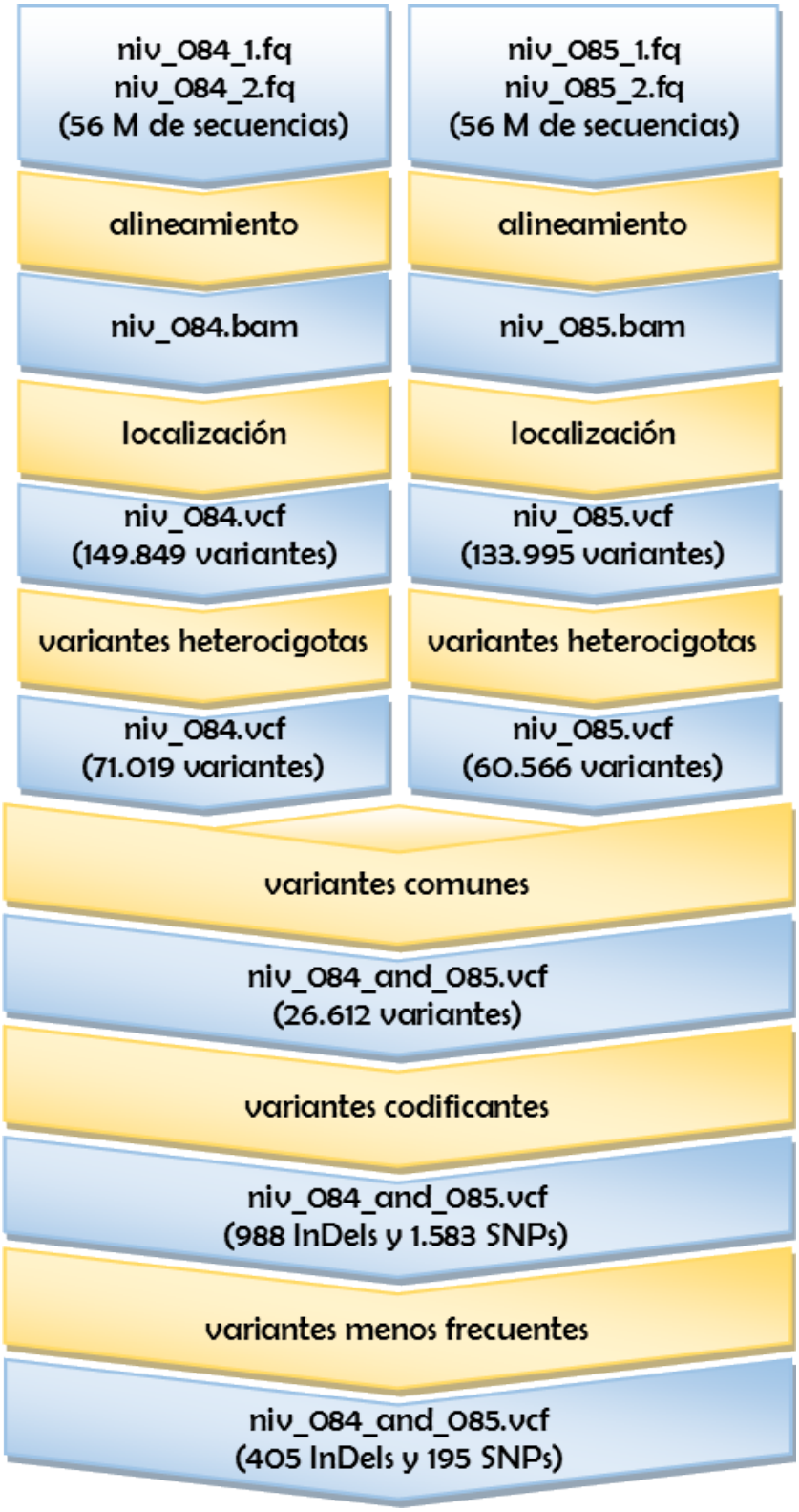


Figura 6.2.: *Ejemplo de flujo de trabajo.*

7

CONCLUSIONES Y TRABAJO FUTURO

Es evidente que con DNAnalytics solo hemos rascado la superficie de algo mucho mayor. El secuenciamiento masivo de ADN representa uno de los grandes logros científicos y tecnológicos del ser humano. Esta tecnología, impensable hace años, unida a la gran potencia informática de la actualidad, nos permite conocer nuestro código genético y dilucidar los miles de genes que codifican tanto nuestra especie como la de otros seres vivos.

Poder determinar aquellas variantes causantes de una enfermedad, o que hacen al individuo más o menos susceptible a un fármaco supone un gran avance en el tratamiento médico, y abre las puertas a un nuevo diagnóstico, que a la larga será más económico y preciso que los sistemas actuales.

Aún quedan muchas cosas por hacer en el flujo de trabajo de la localizacón y filtrado de variantes en el fenotipo, algunas de las cuales son:

- Buscar, gestionar y armonizar nuevas herramientas de filtrado y anotación de variantes.
- Implementar las herramientas más utilizadas en un paquete único e independiente, para aumentar la portabilidad del programa, evitando la dependencia de otros programas.
- Estudiar las regiones del ADN con lecturas bajas durante la secuenciación NGS y que, por lo tanto, no tienen presencia en el flujo de trabajo de la búsqueda de variantes.
- Implementar nuevas herramientas de localización de variantes basadas en el número de lecturas por posición.
- Encontrar, o implementar, nuevos algoritmos para la búsqueda de variantes y el alineamento de secuencias.
- Reducir el tiempo de procesado de datos, con la adaptación del programa ya sea a máquinas supercomputadoras, o a tarjetas gráficas de alto rendimiento.

Parte IV

APÉNDICE

A

FORMATOS DE ARCHIVO

En este apédice se explica brevemente cómo son los formatos de archivos más utilizados en el desarrollo y ejecución de DNnalytics. En la tabla A.1 veremos cuál es el rol de cada uno.

FASTA	Almacena el ADN de referencia.
FASTQ	Almacena las secuencias del ADN de muestra.
SAM/BAM	Almacena las secuencias alineadas.
VCF	Almacena las variantes de un ADN de muestra.

Tabla A.1.: *Formatos más usados.*

A.1 FASTA

Formato basado en texto que se utiliza para representar secuencias de ácido nucleico o proteínas. Se trata del estándar adoptado por la comunidad bioinformática para representar secuencias genéticas de cualquier ser vivo.

Una secuencia FASTA se compone de dos secciones:

- *Línea de descripción*: Comienza con el símbolo >(mayor que), y la palabra siguiente (no debe de haber ningún espacio después del símbolo >) es el identificador de la secuencia. El resto de la línea (opcional) puede contener comentarios o una descripción de la secuencia.

- *Secuencia*: El resto de líneas representan la secuencia, sin espacios. No puede haber líneas en blanco, y la secuencia termina al llegar al final del fichero o el símbolo > al principio de una línea, lo cual significa que comienza una secuencia nueva. Por tradición, cada línea contiene 60 u 80 letras.

Dependiendo de si se quiere representar una cadena de aminoácidos o una cadena de bases nitrogenadas, se restringe el uso de caracteres a los siguientes:

- *Ácido nucleico*: Se representa como una secuencia de bases nitrogenadas. Se permiten los siguientes símbolos:

Símbolo	Significado
A	Adenina
G	Guanina
U	Uracilo

Símbolo	Significado
T	Timina
C	Citosina
N	Cualquiera

Tabla A.2.: *Bases nitrogenadas*

La Tabla A.3 muestra el contenido de un fichero FASTA de bases nitrogenadas.

```
>1 dna:chromosome chromosome:GRCh37:1:1:249250621:1
CTCACGTCACGGTGGCGCGGCGCAGAGACGGGTAGAACCTCAG
ATCGACCGCCCCTTGCTTGCAGCCGGGCACTACAGGACCCGCT
CAGGGCGCCCCCTGCTGGCGACTAGGGCAACTGCAGGGCTCTC
CAGCGCCCCCTGCTGGCGCCGGGGCACTGCAGGGCCCTCTTGC
CACGCCGCCTGCTGGCAGCTAGGGACATTGCAGGGTCCTCTTG
GCACGCCCACCTGCTGGCAGCTGGGGACACTGCCGGGCCCTCT
CGGATTATAGGGAAACACCCGGAGCATATGCTGTTTGGTCTCA
GGATTCCTGGGTTTAAAAGTAAAAAATAAATATGTTTAATTTG
TTTAAACGAGATTGCCAGCACCGGGTATCATTCACCATTTTTC
TCAGCCTTTTCTTTGACCTCTTCTTTCTGTTCATGTGTATTTG
```

Tabla A.3.: *Fichero FASTA de bases nitrogenadas.*

- *Proteína*: Representada como una cadena de aminoácidos.

Símbolo	Significado
A	Alanina
D	Acido aspártico
C	Cisteína
E	Acido glutámico
H	Histidina
L	Leucina
M	Metionina
P	Prolina
T	Treonina
Y	Tirosina
-	Hueco en la secuencia

Símbolo	Significado
R	Arginina
N	Asparagina
Q	Glutamina
G	Glicina
I	Isoleucina
K	Lisina
F	Fenilalanina
S	Serina
W	Triptófano
V	Valina
X	Cualquiera

Tabla A.4.: *Aminoácidos*

La Tabla A.5 muestra el contenido de un fichero FASTA de aminoácidos.

```
>4924382
MHHPILLSAFISIASSLRLNAPTVNKRGSTIAQPGSIKDVIVTSFN
HNLSLKLYNNYNGGPINAYIQGLDSNGAIVFITSNGTLIHPRSNNS
ITTSLTSGRVYFSEGNLKFFTINLGAGDGLVQPSVNNLHDPSASLN
GLILSMMLSTKDGGTPQITRGLRANAVYDLCEGLFKQTANDGYLWL
AADFEDYWQDYVDTVWEYYSSHTLAIDTQTPLGQVECQVTNDTLYC
LQEGDNPVHVAVIPRLCAAFVRSTLLIRGGDVQPRLNSSYHYSVSP
VNPNGHEDVSGLVSSGNPDTLTVYVGGPPN
```

Tabla A.5.: *Fichero FASTA de aminoácidos.*

A.2 FASTQ

Formato basado en texto que representa cadenas de nucleótidos y sus niveles de calidad. Es el formato estándar para almacenar la salida de las máquinas de secuenciación, y es muy similar a su homólogo FASTA.

Las secuencias FASTQ se componen de:

- *Línea de descripción*: Comienza con el símbolo @ (arroba), seguido del identificador y una descripción, como en el formato FASTA.
- *Secuencia*: Líneas sin espacios en blanco, con los caracteres que representan la secuencia.
- *Cabecera de calidades*: Comienza por el símbolo + (más), y puede contener una descripción o no.
- *Niveles de calidad*: Una secuencia de la misma longitud que la cadena superior, donde cada carácter indica el nivel de calidad del elemento situado en la misma posición en la cadena.

Vemos un ejemplo de secuencia FASTQ en la Tabla A.6:

```
@FCC1DYYACXX:6:1101:1701:1935/1
NCAGAGACGCCCCCGCACAGGATCCAGAGCAAGCACCGTC
+
BPYacaccceaegfda^_cdebddfefbee'dghcuyeag
```

Tabla A.6.: *Secuencia FASTQ*

A.3 SEQUENCE ALIGNMENT/MAP

El formato SAM, o BAM (Binary Alignment/Map) cuando está comprimido, se utiliza para almacenar alineamientos de secuencias. Se divide en dos partes:

- *Cabecera*: Puede estar al principio del fichero o en un fichero aparte de mismo nombre y extensión .sai o .bai. Cada línea comienza con el símbolo @ (arroba) seguido de un código. Después, siguiendo el formato ETIQUETA:VALOR, se añaden los campos.

```
@HD VN:1.3 SO:coordinate
@SQ SN:ref LN:45
```

Las cabeceras mínimas requeridas son:

Cabecera	Etiqueta	Descripción
@HD	Línea de cabecera	
	VN	Versión
@SQ	Secuencia de referencia	
	SN	Nombre de la secuencia
	LN	Longitud de secuencia
@RG	Grupo de secuencias	
	ID	Identificador de grupo
	PL	Plataforma
	PU	Unidad de plataforma
	LB	Biblioteca usada
	SM	Muestra
@PG	Programa	
	ID	Identificador de programa

Tabla A.7.: *Cabeceras para los ficheros SAM*

- *Cuerpo*: Cada fila representa una secuencia. Los campos están separados por tabulador.

```
r001  163  ref  7   30  8M2I4M1D3M  =  37  39   ...
r002  0    ref  9   30  3S6M1P1I4M  *  0   0    ...
r003  0    ref  9   30  5H6M        *  0   0    ...
r004  0    ref  16  30  6M14N5M     *  0   0    ...
r003  16   ref  29  30  6H5M        *  0   0    ...
r001  83   ref  37  30  9M          =  7   -39  ...
```

Las columnas significan, de izquierda a derecha:

1	QNAME	Nombre de la muestra. Si todas las secuencias vienen de la misma muestra, este nombre será idéntico. A muestras distintas, nombres distintos.
2	FLAG	Bits de control.
3	RNAME	Nombre de la secuencia de referencia. Este nombre tiene que ser idéntico a una cabecera @SQ.
4	POS	Posición en la referencia de la primera coincidencia con la secuencia.
5	MAPQ	Calidad del alineamiento, basado en la probabilidad de que el alineamiento sea correcto o no.
6	CIGAR	Cadena que indica como se alinea la secuencia a la referencia. 8M2I4M1D3M (8 matches, 2 insertions, 4 matches, 1 deletion 3 matches).
7	RNEXT	Nombre de la siguiente referencia con la que también se alinea la secuencia. Si es la misma, aparece el símbolo = (igual).
8	PNEXT	Posición en la siguiente referencia donde se produce la primera coincidencia.
9	TLEN	Tamaño del fragmento que va desde la primera coincidencia en la referencia hasta la última coincidencia en la misma referencia. Si solo existe una coincidencia, TLEN será igual al tamaño de la secuencia, pero si dentro de la misma referencia coincide varias veces, será mayor, hasta un máximo del tamaño de la referencia. Este número se utiliza para concer la cercanía de los alineamientos de la misma secuencia.
10	SEQ	La secuencia, extraída del fichero FASTQ.
11	QUAL	Los valores de calidad, extraídos del fichero FASTQ.

Tabla A.8.: *Columnas para los ficheros SAM*

A.4 VARIANT CALL FORMAT

El formato VCF[7] es un formato genérico para almacenar información de polimorfismos en el ADN, como SNPs, InDels y variantes estructurales, así como todo tipo de anotaciones. VCF se puede almacenar comprimido y se puede indexar para acceder rápidamente a variantes en un rango de posiciones.

La cabecera contiene un número arbitrario de líneas que comienzan con los caracteres ## (doble almohadilla). Solo la primera, ##fileformat, y la última, #CHROM, son obligatorias. Las cabeceras se utilizan para describir de forma estándar las etiquetas y anotaciones que se utilizan en la sección de datos.

La última línea de la cabecera debe escribirse siempre igual, con los nombres de las columnas de datos separados por tabulador, a saber:

CHROM	Cromosoma
POS	Posición
ID	Identificador único de variante
REF	Alelo de referencia
ALT	Lista de variantes, separadas por coma
QUAL	Valor de calidad
FILTER	Filtros aplicados
INFO	Lista de anotaciones, separadas por ; (punto y coma).
FORMAT	[Opcional] Formato de la siguiente columna, etiquetas separadas por : (dos puntos)
SAMPLE1	[Opcional] 1 o más columnas con la información representada en FORMAT para distintas muestras.

Tabla A.9.: *Columnas del formato VCF.*

Así es un fichero VCF.

```
##fileformat=VCFv4.1
##fileDate=20090805
##source=myImputationProgramV3.1
##phasing=partial
##INFO=<ID=DP,Number=1,Type=Integer,Description="Total Depth»
##FILTER=<ID=s50,Description="Less than 50% have data»
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype»
#CHROM  POS      ID         REF  ALT  QUAL  FILTER  ...
20      14370    rs6054257  G    A    29    PASS    ...
20      17330    .          T    A    3     q10     ...
20      1110696  rs6040355  A    G,T  67    PASS    ...
20      1230237  .          T    .    47    PASS    ...
```

Tabla A.10.: *Ejemplo de fichero VCF.*

BIBLIOGRAFÍA

[1] URL: http://www.ncbi.nlm.nih.gov/projects/genome/assembly/grc.

[2] URL: http://www.oracle.com/technetwork/es/java/javafx/overview/index.html.

[3] McKenna A y col. "The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data". En: *Genome Res.* 20.9 (sep. de 2010), págs. 1297-1303.

[4] Ivan A. Adzhubei, Steffen Schmidt, Leonid Peshkin, Vasily E. Ramensky, Anna Gerasimova, Peer Bork, Alexey S. Kondrashov y Shamil R. Sunyaev. "A method and server for predicting damaging missense mutations". En: *Nat Methods* 7.4 (abr. de 2010), págs. 248-249.

[5] CA Albers, G Lunter, Daniel G MacArthur, Gilean McVean, Willem H Ouwehand y Richard Durbin. "Dindel: Accurate indel calls from short-read data". En: *Genome Research* 21 (oct. de 2010), págs. 961-973.

[6] The 1000 Genomes Project Consortium. "A map of human genome variation from population-scale sequencing". En: *Nature* 467 (oct. de 2010), 1061–1073.

[7] Petr Danecek y col. "The variant call format and VCFtools". En: *Bioinformatics* 27.15 (ago. de 2011), págs. 2156-2158.

[8] McKusick-Nathans Institute of Genetic Medicine Johns Hopkins University (Baltimore MD). *Online Mendelian Inheritance in Man, OMIM®*. URL: http://omim.org/.

[9] Li H., Handsaker B., Wysoker A., Fennell T., Ruan J., Homer N., Marth G., Abecasis G., Durbin R. y 1000 Genome Project Data Processing Subgroup. "The Sequence alignment/map (SAM) format and SAMtools". En: *Bioinformatics* 25 (2009), págs. 2078-2079.

[10] The Broad Institute. *GATK Main Page*. URL: http://www.broadinstitute.org/gatk/.

[11] García-Villarreal L, Daniels S, Shaw SH, Cotton D, Galvin M, Geskes J, Bauer P, Sierra-Hernández A, Buckler A y Tugores A. "High prevalence of the very rare Wilson disease gene mutation Leu708Pro in the Island of Gran Canaria

(Canary Islands, Spain): a genetic and clinical study". En: *Hepatology* 32.6 (dic. de 2000), págs. 1329-1336.

[12] Peña-Quintana L, García-Luzardo MR, García-Villarreal L, Arias-Santos MD, Garay-Sánchez P, Santana A, González-Santana D, Ramos-Varela JC, Rial-González R y Tugores A. "Manifestations and evolution of Wilson disease in pediatric patients carrying ATP7B mutation L708P". En: *J Pediatr Gastroenterol Nutr* 54.1 (ene. de 2012), págs. 48-54.

[13] Ben Langmead y Steven L Salzberg. "Fast gapped-read alignment with Bowtie 2". En: *Nature Methods* (abr. de 2012), 357–359.

[14] Heng Li y Richard Durbin. "Fast and accurate long-read alignment with Burrows-Wheeler transform". En: *Bioinformatics* 26 (2010). PMID: 20080505, págs. 589-595.

[15] Ruiqiang Li, Yingrui Li, Karsten Kristiansen y Jun Wang. "SOAP: short oligonucleotide alignment program". En: *Bioinformatics* 24.5 (ene. de 2008), págs. 713-714.

[16] Josep Mª. Barres Manuel, Ramón Forés Rupérez y María del Pilar González de la Fuente. *Gran Vox*. 5.ª ed. Vol. Biología. Biblograf, dic. de 1998.

[17] Encyclopedia of Mathematics. *Bayesian approach*. URL: http://www.encyclopediaofmath.org/index.php?title=Bayesian_approach&oldid=29437.

[18] Antonio Medrano. *Gran Referencia Anaya*. 1.ª ed. Vol. 14. Biblograf, 2000.

[19] Pauline C. Ng y Steven Henikoff. "SIFT: predicting amino acid changes that affect protein function". En: *Nucleic Acids Res* 31.13 (jul. de 2003), págs. 3812-3814.

[20] *Picard*. URL: http://picard.sourceforge.net/.

[21] Peter J. Russell. *Genetics*. 4.ª ed. Harper Collins, 1996.

[22] Miller SA, Dykes DD y Polesky HF. "A simple salting out procedure for extracting DNA from human nucleated cells". En: *Nucleic Acids Research* 16.3 (feb. de 1988), pág. 1215.

[23] R M Schaaper. "Base selection, proofreading, and mismatch repair during DNA replication in Escherichia coli". En: *The Journal of Biological Chemistry* 268 (nov. de 1993), págs. 23762-23765.

[24] *VCFTools*. URL: http://vcftools.sourceforge.net/.

[25] McLaren W, Pritchard B, Rios D, Chen Y, Flicek P y Cunningham F. "Deriving the consequences of genomic variants with the Ensembl API and SNP Effect Predictor". En: *Bioinformatics* 26.16 (2010), págs. 2069-2070.