Memoria del Trabajo de Fin de Título del Grado de Ingeniería en Informática de la Universidad de Las Palmas de Gran Canaria

TÍTULO

Combinación y anotación de variantes genéticas

AUTOR

Jacob Henríquez González

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

Junio de 2017

AGRADECIMIENTOS

Quiero dar las gracias a todas las personas que han hecho posible el desarrollo de este Trabajo de Fin de Título:

A mi tutor, Antonio Tugores, por darme la oportunidad de participar en un proyecto de investigación real y por proporcionarme todos los conocimientos necesarios sobre biología y genética sin los que este trabajo no habría podido llevarse a cabo, y por supuesto, sin los que no habría podido redactar este documento. También a mi tutora, Francisca Quintana, por su apoyo en la realización de este proyecto, paciencia y ayuda en la corrección de la presente memoria.

A Pascual Lorente, compañero de la Unidad de Investigación, por toda la ayuda y consejos durante el desarrollo del proyecto. También a todos los compañeros de la Unidad de Investigación por hacerme sentir como uno más desde el primer día y que mi estancia allí fuese una experiencia muy agradable que quedará siempre en el recuerdo.

A mis compañeros de clase, con los que pasé muchas tardes de estudio y trabajo, así como muchos momentos divertidos.

A mis profesores, por todos los conocimientos y habilidades que me han ayudado a conseguir durante los estudios universitarios.

A mi familia, por todo su apoyo tanto emocional como económico, y por darme fuerzas en aquellos momentos en los que sentía que no podía más. Especialmente, agradecer a mi madre, María, por darme ánimos y fuerzas en estos últimos años en los que he pasado por una situación personal complicada. Gracias por ayudarme a seguir avanzando.

Muchas gracias a todos.

Jacob Henríquez González

RESUMEN

La búsqueda de variantes genéticas es de especial importancia en la medicina para determinar el origen de enfermedades mendelianas y, además establecer tratamientos personalizados, ya que la respuesta de un individuo a un medicamento puede estar determinada por su configuración genética. Una estrategia para la búsqueda de estas variantes es la secuenciación del exoma completo que, aunque cubre la mayor parte de las secuencias que codifican las proteínas, tiene una tasa de error de identificación de variantes que es incompatible con una funcionalidad diagnóstica.

Con este Trabajo de Fin de Título en la ***Unidad de Investigación del Complejo Hospitalario Universitario Insular - Materno Infantil* (*UICHUIMI*)** se pretende desarrollar las herramientas ***Combinator*** y ***False Positive Searcher***, para añadir las funcionalidades de combinación de variantes y detección de falsos positivos a la aplicación global, *Combination and Annotation Tool* (*COAT*), desarrollada y utilizada en la UICHUIMI para el análisis de la secuenciación del exoma completo.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE CÓDIGOS

1. INTRODUCCIÓN

1.1 EL ADN Y LAS ENFERMEDADES GENÉTICAS

El *ácido desoxirribonucleico* (*ADN*) almacena la información genética necesaria para definir, en su mayor parte, la estructura y el funcionamiento de los organismos vivos, y es responsable de su transmisión hereditaria. La totalidad de la información genética que posee un organismo se denomina *genoma*[8], es decir, el conjunto de *genes*[7] donde están contenidas la mayoría de las claves para la diferenciación y el funcionamiento de las células que forman los distintos tejidos y órganos de un individuo.

El genoma codifica una serie de características o rasgos de un organismo, como su morfología, desarrollo, propiedades bioquímicas, fisiología y comportamiento. A todas estas características las denominamos *fenotipo*, y es la representación observable de nuestra información genética, el *genotipo*.

Como vemos en la figura 1.1, el ADN está formado por dos largas cadenas (complementarias, 3' y 5') de *nucleótidos*[12] unidas entre sí formando una doble hélice. Se mantienen unidas mediante enlaces entre las bases nitrogenadas de ambas cadenas.

Las cuatro bases nitrogenadas que se encuentran en el ADN son: *adenina (A)*, *timina (T)*, *citosina (C)* y *guanina (G)*. Los enlaces entre las bases se realizan por medio de puentes de hidrógeno, de forma que la adenina se une con la timina, y la citosina con la guanina. Esta estructura es responsable de que la molécula de ADN sea un polímero[13] robusto y flexible a la vez, mientras que la complementariedad de las bases asegura que cada molécula contenga dos copias de la información: en la cadena codificante y en la complementaria.

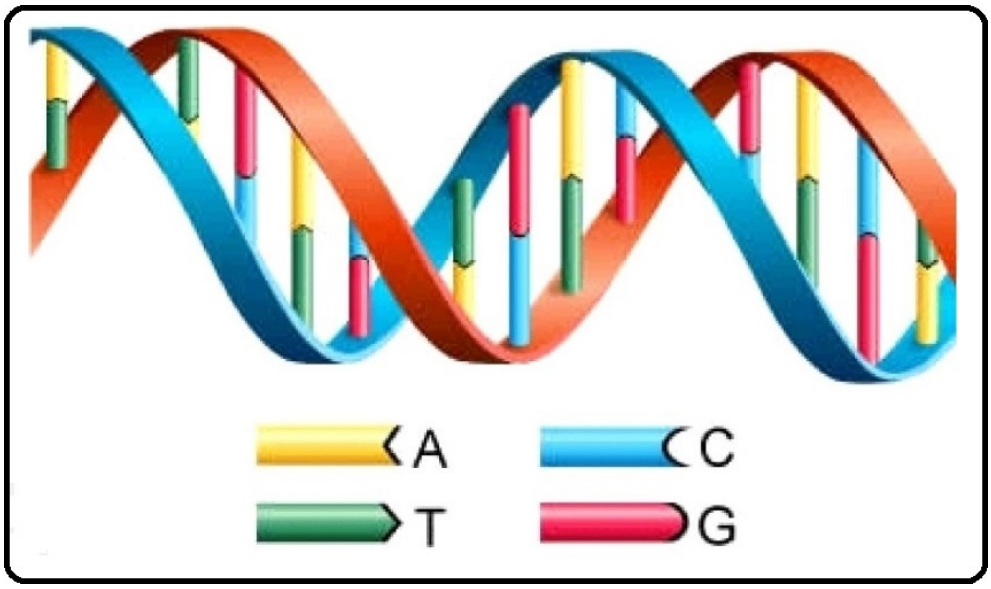

Figura 1.1: e*structura del ADN.*

En concreto, en el ser humano tenemos 46 cadenas de ADN, que se organizan en estructuras denominadas *cromosomas*, distribuidas en 23 parejas, de forma que 23 cromosomas se heredan de la madre y los otros 23 del padre. De estas parejas, 22 son homólogas, es decir, tienen el mismo tamaño y estructura. El último par de cromosomas es el que determina el sexo de la persona (cromosomas sexuales), siendo homólogo en las mujeres (XX) y heterólogo en los hombres (XY).

Existen aproximadamente unos 20000 genes [1], que son aquellas regiones del ADN que codifican *proteínas*[14]. Las proteínas desempeñan una gran variedad de funciones en la célula, incluidas estructurales, mecánicas y bioquímicas. El proceso de creación de proteínas a partir del ADN se denomina *síntesis de proteínas*.

La presencia de *variantes*[20] *genéticas* puede provocar cambios en la estructura final de la proteína, lo que puede ser origen de una función diferente que podría desembocar en enfermedad. Estas variantes se pueden heredar de los progenitores en el momento en que se genera el ADN del

nuevo individuo, pero también pueden surgir variantes espontáneas, siendo estas últimas mucho menos frecuentes.

Existen variantes del tamaño de un gen, del tamaño de un cromosoma y variantes puntuales que afectan a uno o varios pares de bases nitrogenadas. Las variantes puntuales podemos clasificarlas de la siguiente manera:

- *Single Nucleotide Polymorphism (SNP):* se sustituye una base nitrogenada por otra sin modificar el tamaño de la cadena. Por ejemplo:

```
Antes de la variación:    ATCGA
Después de la variación:  ATGGA
```

Código 1.1: e*jemplo de SNP en el que la base C ha sido sustituida por la base G.*

- *Insertion/Deletion (InDels):* una o varias bases nitrogenadas son insertadas o eliminadas en un punto concreto de la cadena, alterando su tamaño. Por ejemplo:

```
Antes de la inserción:    ATCGA
Después de la inserción:  ATCAGA
```

Código 1.2: e*jemplo de inserción (insertion). Se inserta una base A entre las bases C y G.*

```
Antes de la eliminación:    ATCGA
Después de la eliminación:  AT-GA
```

Código 1.3: e*jemplo de eliminación (deletion). Se elimina la base C situada entre las bases T y G.*

Consideramos que una variante es una *mutación* cuando el efecto en el individuo es negativo. Una consecuencia de las mutaciones son precisamente las enfermedades genéticas, que pueden estar provocadas por una o unas pocas variantes.

No todas las variantes contenidas en el ADN se expresan en el fenotipo. Gregor Mendel, considerado el padre de la genética moderna, estableció tres leyes para determinar la forma en que se transmiten las características en cada generación. Para ello, cosechó guisantes hasta obtener razas puras, es decir, 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:

- *1ª ley de Mendel:* al cruzar entre sí dos razas puras se obtiene una generación que es idéntica entre sí e idéntica a uno de los padres (ver figura 1.2).

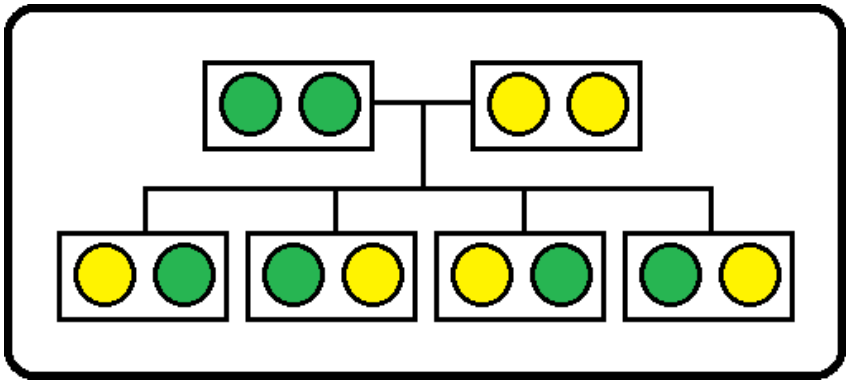

Figura 1.2: *primera ley de Mendel.*

Todos los guisantes cosechados eran de color amarillo, con lo cual llegó a la conclusión de que existen características que son *dominantes* y características que son *recesivas*[16]. Los hijos son idénticos al padre que tenga una característica dominante, lo que puede dar a pensar que la característica recesiva desaparece, aunque Mendel no lo creía así. Por eso, volvió a cruzar los guisantes amarillos que obtuvo (híbridos).

- *2ª ley de Mendel:* 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 (ver figura 1.3).

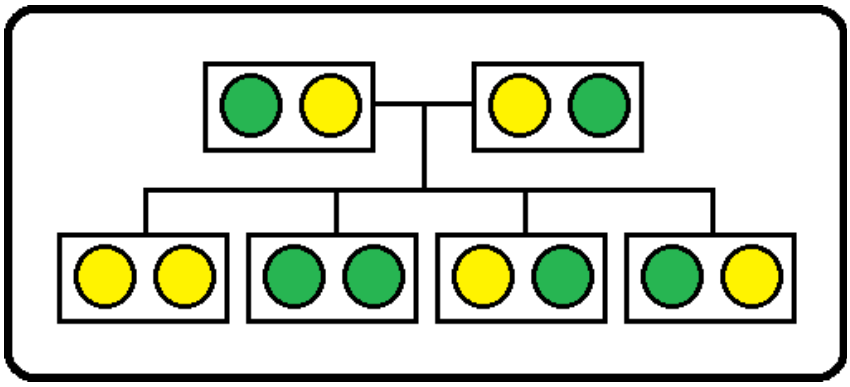

Figura 1.3: *segunda ley de Mendel.*

En esta nueva generación obtuvo guisantes verdes y guisantes amarillos, demostrando así 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 varias características distintas para observar la dependencia entre ellas, por ejemplo: color de la semilla, color de la flor o rugosidad de la semilla.

- *3ª ley de Mendel:* al cruzar entre sí dos híbridos para diferentes características, los caracteres hereditarios se separan, ya que son independientes y se combinan entre sí de todas las formas posibles.

Los experimentos de Mendel han servido para explicar el comportamiento de la *herencia* del fenotipo, así como la presencia de los dos *alelos*[1] en cada individuo.

Como ya hemos visto, se calcula que el genoma humano contiene unos 20000 genes, considerando como tales aquellas regiones del genoma que se transcriben para generar moléculas de ARN que pueden tener una funcionalidad por sí mismas (lnRNA, miRNA) o, en su mayoría, convertirse en ARNs mensajeros que luego se traducen en proteínas en los ribosomas[17]. De media, los genes tienen un tamaño de unos 3000 nucleótidos o bases. Teniendo en cuenta que el total del genoma asciende a más de 3000 millones de bases, la suma de todos los genes representa un 2% del genoma humano, que se organiza de forma modular en estructuras que denominamos *exones*[7]. Los exones son seleccionados entre el ADN no codificante para ensamblar los ARNs mensajeros específicos para cada gen y que resultarán en una o varias proteínas codificadas desde un solo gen. El conjunto de exones en el genoma se denomina *exoma*.

El genoma de un individuo contiene cientos de miles de variantes genéticas, la mayoría heredadas de sus progenitores. Estos cambios ocurren, en su mayor parte, en zonas intergénicas de función desconocida. Al reconocer más fácilmente el impacto de los cambios en los genes que tienen función conocida, el análisis específico de las variantes contenidas en el exoma permite concentrar el análisis en ese 2% del genoma, simplificando significativamente la tarea.

Habitualmente nuestro exoma contiene miles de cambios que, en su mayor parte, no tienen una repercusión clínica (no siempre causan una enfermedad). Por ello, es imprescindible filtrar esos cambios encontrados para identificar aquellos que están relacionados con el fenotipo que, en este caso, es la enfermedad que se trata de diagnosticar.

Para reducir la cantidad de variantes a analizar, se recurre al estudio del exoma de los padres o familiares con la misma enfermedad y se seleccionan las variantes comunes entre ellos. También se emplean

controles, que pueden ser familiares o no familiares no afectados por la enfermedad, para descartar variantes que no provocan la afección.

La ventaja de utilizar a un familiar como control es que permite eliminar un mayor número de variantes, aunque se debe tener cuidado con las enfermedades recesivas, es decir, aquellas que no se manifiestan en la persona utilizada como control pero que es portadora.

1.2 ESTADO ACTUAL

La *Unidad de Investigación del Complejo Hospitalario Universitario Insular – Materno Infantil (UICHUIMI)* lleva desde hace años tratando de descubrir el origen de enfermedades genéticas en la población local, con el fin de diagnosticar y proponer un tratamiento adecuado a las mismas. Uno de sus mayores logros ha sido el de identificar una *mutación endémica*[11] en la población afectada por la *enfermedad de Wilson*[4] en la isla de Gran Canaria.

Para identificar las variantes genéticas responsables de varias *enfermedades mendelianas*[5] no caracterizadas en Gran Canaria, la UICHUIMI trabaja con la *secuenciación de ADN*[18] por *Next Generation Sequencing (NGS)* [3], que cubre con lecturas muy fiables un alto porcentaje del exoma. De las miles de variantes que posee cada ser humano, el mayor reto consiste en identificar aquellas sospechosas de producir la enfermedad de objeto del estudio. Este proceso es difícil, ya que la selección implica un proceso muy manual y basado en el conocimiento que se tiene de la patología en cuestión. Por ello, para minimizar el número de variantes sospechosas, es fundamental optimizar al máximo el proceso de filtrado que busca eliminar las variantes presentes también en individuos no afectados, y aquellas que corresponden a errores tecnológicos, que son falsos positivos.

A continuación detallamos las fases seguidas para el análisis del ADN, desde el estudio de una posible hipótesis hasta que se adquieren variantes que pueden originar enfermedades.

Fase 1: planteamiento de la hipótesis.

Se hace un estudio de la familia del paciente o los pacientes susceptibles de padecer una enfermedad de origen genético. Para ello se elabora un árbol genealógico (ver figura 1.4) donde se describe quién tiene la enfermedad y el parentesco que guarda con el resto. Entonces, se plantean una o varias hipótesis que cubran todos los casos posibles acerca del tipo de variante genética esperada: dominante, recesiva o ligada a los cromosomas sexuales. En el apartado 1.3 podemos ver un ejemplo de árbol familiar anotado.

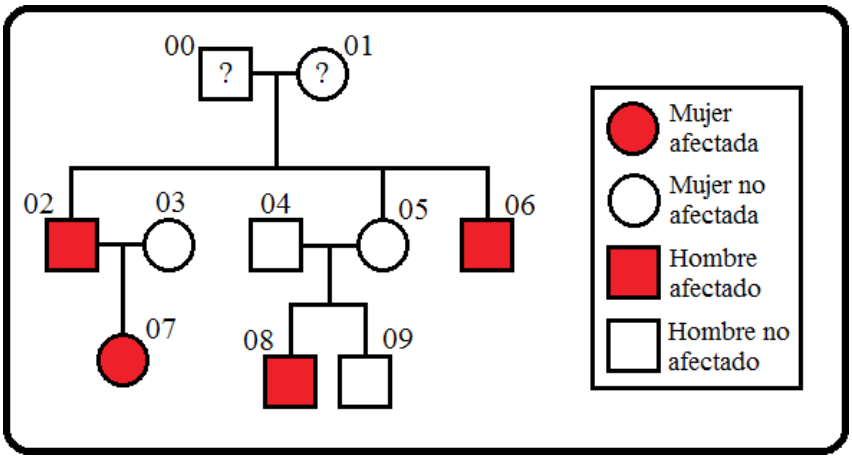

Figura 1.4: *ejemplo de árbol genealógico.*

En la figura 1.4 vemos un ejemplo de árbol genealógico en el que desconocemos la información para 00 y 01, pero sí tenemos datos del resto de individuos. Si nos fijamos en 08, su madre (05) no está afectada por la enfermedad pero sus tíos (02 y 06) sí están afectados. En este caso, podríamos plantear la hipótesis de que la enfermedad es recesiva, lo que implica que 05 sería una portadora *heterocigota*[9] y los afectados 02, 06 y 08

serían *homocigotos*[10] para esa misma mutación. Así pues, podríamos buscar, por ejemplo, las variantes comunes entre 08, 05 y 02.

No obstante, la hipótesis más plausible para este árbol familiar es que es una enfermedad dominante, en la que 05 es portadora asintomática, lo cual es común en este tipo de herencia. No podemos descartar en este caso si la mutación está en los autosomas (cromosomas no sexuales) o en el cromosoma X, ya que el árbol es compatible con ambos escenarios: 07 (XX) lleva el cromosoma X de 02 (XY), 08 (XY) lleva un cromosoma X de 05 (XX) y 06 (XY) lleva un cromosoma X de 01 (XX).

Una vez establecida la hipótesis más probable, se elabora la planificación para las fases posteriores, decidiendo los miembros de la familia que hay que secuenciar y cómo se deberían analizar los datos obtenidos.

Fase 2: obtención del ADN.

A los pacientes para los que queremos secuenciar el ADN se les extrae una pequeña muestra de unos 10 mililitros de sangre utilizando una simplificación del método de precipitación con sales descrito por Miller [4].

Fase 3: secuenciación del ADN.

Las muestras de ADN puro se envían a un laboratorio externo, como *Beijing Genomics Institute (BGI)* [5] o el *Centro Nacional de Análisis Genómico (CNAG)* [6], que se encarga de la secuenciación por NGS. Allí se fragmenta el ADN en millones de trozos de entre 150 y 200 pares de bases, se separan los fragmentos de menor calidad y aquellos que no pertenecen al exoma por medio de técnicas físico-químicas, y se secuencian en un sistema Illumina HiSeq en ambas direcciones, obteniéndose información del mismo fragmento en ambos sentidos.

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 en *formato FASTQ* [7], cada uno representando todas las secuencias leídas en ambas direcciones (3' y 5'), así como un valor de calidad para cada lectura.

Fase 4: alineamiento de las secuencias.

Los archivos (FASTQ) normalmente contienen de 25 a 60 millones de secuencias con longitudes de aproximadamente 90 pares de bases, por lo que almacenan unos 3000 millones en su totalidad.

El problema que existe es que la información que aportan estos ficheros no determina las posiciones en las que se encuentran las secuencias, por lo que es necesario un proceso de alineamiento para situar cada secuencia en su posición exacta o aproximada respecto a un *genoma de referencia*, publicado por *The Genome Reference Consortium* [8].

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

- *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.

- *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.

Hemos de tener en cuenta la dificultad de alinear las secuencias cuando existe una variante, ya que no coincide con ninguna región exacta del genoma de referencia e incluso puede coincidir con otra región diferente.

Para el alineamiento de secuencias tenemos disponibles diferentes programas, como por ejemplo: *Burrows-Wheeler Aligner (BWA)* [9], *Bowtie 2* [10] o *Short Oligonucleotide Analysis Package (SOAP)* [11].

Como salida de los programas de alineamiento *BWA* y *Bowtie 2* obtendremos archivos en formato *Sequence Alignment/Map Format* (*SAM)* [12], que son compatibles con varios buscadores de variantes. Por otro lado, el formato de salida de *SOAP* es propio e incompatible con otros programas.

Fase 5: localización de variantes.

Una vez que se han alineado las secuencias, habrá que ir desde el principio hasta el final del genoma de referencia comparándolo con el exoma reconstruido, para identificar las posiciones en las que es diferente. Este proceso es complejo, ya que normalmente en cada posición se encuentran varias lecturas alineadas y pueden diferir.

Antes de realizar el proceso de localización de variantes se refinan los alineamientos. Utilizando *Genome Analysis Toolkit (GATK)* [13] se realinean los vacíos en las lecturas para facilitar la detección de InDels, se marcan lecturas repetidas que no añaden información y se recalibran con un modelo estadístico los valores de calidad. Los ficheros obtenidos de esta herramienta están en formato *Variant Call Format* (*VCF)* [12].

Fase 6: anotación de variantes.

En esta fase se tiene un fichero con miles de variantes, en el cual el objetivo es identificar unas cuantas entre todas ellas. Para ello se hacen una serie de filtros con respecto a ciertos criterios.

La información sobre la ubicación de las variantes es pobre ya que solamente indica la posición en la que se ha producido, por lo que es necesario agregar más información. Los campos más frecuentes que se añaden son:

- *Posición:* indica la pertenencia y posición de la variante en un *codón*[3] determinado y qué nucleótido había en ese mismo codón 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, es decir, se sigue transcribiendo la misma proteína.

- *Peligro:* dentro de las variantes no sinónimas, existen las toleradas y las peligrosas en función de la predicción de lo que podría suponer la sustitución de un aminoácido por otro en esa posición de la proteína. Hay varios algoritmos que utilizan esta información, además del grado de conservación de un aminoácido

determinado en la misma proteína (o similar) en otras especies, para determinar si un cambio no sinónimo podría ser tolerable o no. Entre estos algoritmos encontramos *SIFT* [17] y *Polyphen* [15].

- *Conocimiento:* lista de bases de datos y publicaciones donde se encuentra la variante.

- *Frecuencia:* frecuencia de la variante en las distintas bases de datos consultadas. Normalmente se clasifica por regiones geográficas (Europa, África, Asia...), de forma que podemos saber en qué partes del mundo es más frecuente. La base de datos más popular es *1000 Genomes* [14].

Existen diferentes herramientas anotadoras, como por ejemplo: *Variant Effect Predictor (VEP)* [16], *Sort Intolerant From Tolerant (SIFT)* [17] o *Annotate Variation (ANNOVAR)* [18]. Aunque muchas de las herramientas pueden ejecutarse localmente, es recomendable utilizar el servicio web (para las que lo tengan disponible), ya que los resultados siempre están actualizados.

Fase 7: filtrado.

Esta fase es la más importante ya que en la localización de variantes (fase 5) se identifican unos cientos de miles de variantes. La finalidad es hacer un filtrado para destacar las variantes que puedan ser sospechosas de causar una enfermedad.

Para ello se filtra por frecuencias, suprimiendo las variantes que contengan una frecuencia muy alta (por ejemplo, mayor que 0.01 para una enfermedad dominante o mayor que 0.005 para una enfermedad recesiva), ya que se considera que por encima de estas frecuencias perjudicarían a demasiadas personas. Por otra parte, si existen varias variantes en la misma

posición, se descartan aquellas con un bajo número de lecturas menores y con un bajo valor de calidad.

Por último se clasifican las variantes según la hipótesis acerca de su tipo de herencia en *homocigotos* o *heterocigotos*, dependiendo de si el modelo planteado es recesivo o dominante respectivamente. También se pueden buscar variantes comunes entre diferentes pacientes con una misma enfermedad.

1.3 BÚSQUEDA DE VARIANTES COMUNES

Tal y como vimos en el apartado 1.2 (fase 1), antes de analizar las posibles variantes que puedan presentar los pacientes, se prepara una hipótesis mediante la elaboración de un árbol genealógico, para hacer un estudio de la familia del paciente con el fin de identificar en el árbol a los individuos que padecen o que son portadores de una enfermedad genética.

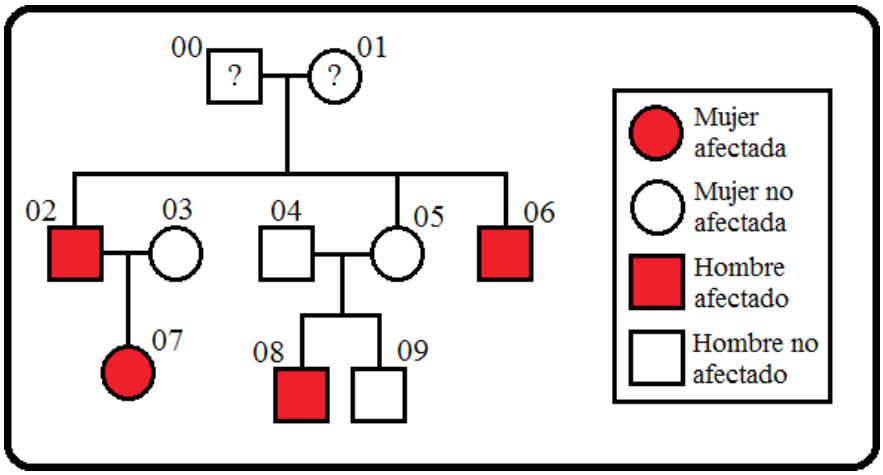

Figura 1.4: *ejemplo de árbol genealógico.*

Partiendo de la figura 1.4 y basándonos en lo que hemos visto anteriormente (ver apartado 1.2, fase 1), en el caso de una enfermedad dominante, 02, 07, 08, 05 y 06 deberían ser todos portadores de la variante, y suponemos que son todos heterocigotos. En el caso de que sospecháramos

que fuera recesiva, 02, 07, 08 y 06 tendrían que ser homocigotos (1/1) si tuviéramos suficiente evidencia de endogamia (por ejemplo, si 00 y 01 son hermanos o primos hermanos), mientras que 05 tendría que ser heterocigota (0/1) en esa misma variante.

1.4 ZONAS DE BAJA COBERTURA

Una *zona de baja cobertura* es una región del exoma con una frecuencia de lectura baja.

La secuenciación del ADN cubre la mayor parte de los exones del genoma, pero no localiza algunas regiones, lo que imposibilita la detección de variantes en ellas. De esta forma, el poder de la herramienta de secuenciación se ve limitado para detectar mutaciones, ya que puede haber genes que hayan quedado sin analizar.

Esto se traduce en que cuando tenemos un fichero con un listado de variantes, desconocemos si las variantes que no figuran en el mismo no existen realmente, o si por el contrario no aparecen porque desconocemos lo que hay en esa región.

Por ello, en la UICHUIMI han desarrollado una herramienta que les permite la detección de regiones del exoma con poca cobertura. Dicha utilidad, denominada *Missing Sequencing Tools (MIST)* [19], ofrece al usuario una lista de regiones donde la frecuencia de lectura no supera un cierto umbral establecido por el usuario.

A la hora de buscar variantes comunes en diferentes pacientes, resulta de gran utilidad conocer las regiones de baja cobertura puesto que puede darse el caso en el que un individuo tiene una mutación, pero la variante no aparece en su fichero correspondiente por encontrarse en una zona con una baja frecuencia de lectura, es decir, un falso negativo.

1.5 VALIDEZ DE LOS DATOS

Hemos de tener en cuenta la existencia de errores en la obtención de los datos que pueden llevar a la aparición de variantes no reales (*falsos positivos*), o a la no aparición de variantes reales (*falsos negativos*), siendo más peligroso el primer caso porque se nos indica la presencia de una variante cuando realmente no la hay.

Como hemos visto, un falso negativo es la variante real que se pierde porque la zona en la que se encuentra localizada no ha sido suficientemente secuenciada. Estas zonas son precisamente las que identifica *MIST* [19] (ver apartado 1.4).

Por otro lado, un falso positivo lo encontramos en zonas con buena calidad de lectura pero que realmente son un error. Los falsos positivos pueden deberse, por ejemplo, a fallos químicos en los procesos de laboratorio, errores de secuenciación, regiones duplicadas en las muestras, errores de alineamiento o muestras desalineadas [20].

Nuestra meta es poder identificar los falsos positivos puesto que los falsos negativos ya se han perdido al estar en zonas no secuenciadas. Por lo tanto, necesitamos una herramienta que consiga identificar y eliminar los falsos positivos que aparecen en las muestras.

Para este proyecto, nos hemos centrado en la detección de *falsos positivos de tipo solapamiento*.

Un falso positivo de tipo solapamiento se puede producir cuando en medio de una cadena donde se repite la misma base o nucleótido, aparece una base distinta. Veamos a continuación un ejemplo que ilustre el problema:

```
... C C C C C C C C C C A C C C C C C C C C C ...
```

Código 1.4: *ejemplo de secuencia en la que se lee una ristra de bases C, a continuación una base A, y nuevamente una serie de bases C.*

Como podemos observar en el código 1.4, tenemos una secuencia en la que se lee una serie de bases C, pero en cierto punto de la misma nos encontramos una base A. Esto puede ocasionar que se produzca un error de lectura en el que se da por hecho que hay una base C en la posición en la que realmente tenemos la base A. Es decir, se produce un solapamiento en el proceso de lectura en el que el elemento distinto (*base A*) se toma como si fuese igual al resto (*bases C*).

Por otro lado, hemos de tener en cuenta que para una secuencia se llevan a cabo varias lecturas que dan soporte a la base que se toma como válida. Una base correspondiente a una posición concreta será leída varias veces con el objeto de tener una mayor seguridad de que sea correcta. En el caso de nuestra secuencia de ejemplo (código 1.4), supongamos que la posición en la que se encuentra la base A ha sido leída 22 veces: 15 lecturas para A y 7 lecturas para C.

Finalmente, para esta posición, la herramienta que se encarga del proceso de identificación puede determinar que la base existente es C, a pesar de que realmente tenemos una base A. Esto es debido precisamente al error de lectura (solapamiento) que se puede dar en la situación que ilustrábamos en el código 1.4.

Además, cada lectura que da soporte a la base tiene asociada una calidad de lectura. En el caso que nos ocupa, las lecturas que soportan que la base encontrada es C tienen una calidad baja en comparación a las lecturas que soportan que la base es A.

1.6 OBJETIVOS DEL TFT

Debido a la gran cantidad de datos que se genera en el estudio del genoma humano, se necesitan herramientas software que los procesen de forma automática y presenten al usuario final resultados cortos e informados, facilitando el procesado y la manipulación de los datos.

Con este Trabajo de Fin de Título se pretende desarrollar un software capaz de cubrir los siguientes objetivos:

1. Combinar variantes de varias muestras.

1.1 Buscar variantes comunes en los pacientes seleccionados.

1.2 Excluir variantes de personas no afectadas (controles).

2. Incorporar a la herramienta de combinación la funcionalidad de identificar variantes que se encuentran en una zona de baja cobertura (aplicación MIST desarrollada en la UICHUIMI).

3. Anotar las variantes resultantes (añadir la información obtenida de uno o varios anotadores).

4. Detectar las variantes que puedan ser *falsos positivos (de tipo solapamiento)*.

2. COMPETENCIAS ESPECÍFICAS CUBIERTAS

Para la realización de este Trabajo de Fin de Título se requieren una serie de competencias, de las cuales, enumeraremos y justificaremos las siguientes:

Competencia CII01: capacidad para diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad, conforme a principios éticos y a la legislación y normativa vigente.

Se ha diseñado y desarrollado una serie de herramientas para las que se ha tenido que hacer un estudio previo sobre un conjunto de ficheros de entrada, en diferentes formatos y que contienen distintos tipos de información genética, con el fin de cumplir con los objetivos del TFT (ver apartado 1.6). Dado el carácter personal de esta información genética obtenida de diferentes pacientes, su utilización se ha hecho conforme a la Ley de Protección de Datos (LOPD) (ver apartado 6).

Se han incluido también los recursos necesarios utilizados (ver apartado 4.2).

Además, para cada herramienta se ha llevado a cabo una serie de pruebas para evaluar su funcionamiento.

Competencia CII02: capacidad para planificar, concebir, desplegar y dirigir proyectos, servicios y sistemas informáticos en todos los ámbitos, liderando su puesta en marcha y su mejora continua y valorando su impacto económico y social.

Para el desarrollo de este proyecto se ha seguido una metodología de trabajo determinada (ver apartado 4.1) con el fin de ir abarcando cada uno de los objetivos perseguidos (ver apartado 1.6).

En nuestro caso particular, se trata de un proyecto que se encuentra dentro del ámbito de la investigación, pero que puede tener en un futuro un importante impacto económico y social ya que está destinado principalmente a la mejora en el diagnóstico de enfermedades genéticas (ver apartado 3).

Competencia CII18: conocimiento de la normativa y la regulación de la informática en los ámbitos nacional, europeo e internacional.

Para el desarrollo tanto de este TFT en particular, como de cualquier proyecto, los desarrolladores deben estar al tanto de la normativa y la regulación de la informática en el ámbito nacional, de la Unión Europea, e internacional.

Al tratar con información genética de diferentes pacientes, la UICHUIMI tiene cubierto por su propia estructura el cumplimiento de la Ley de Protección de Datos (ver apartado 6).

Competencia TFG01: ejercicio original a realizar individualmente y presentar y defender ante un tribunal universitario, consistente en un proyecto en el ámbito de la Ingeniería Informática de naturaleza profesional en el que sinteticen e integren las competencias adquiridas en las enseñanzas.

Esta competencia quedará cubierta en el momento en que se haya realizado la presentación de este TFT ante los miembros del tribunal asignados al mismo.

3. APORTACIONES

Con la búsqueda de variantes genéticas se pretende llegar a prevenir o retrasar la aparición de una enfermedad (incluso desde antes del nacimiento), permitiendo tomar decisiones preventivas que mejoren la calidad de vida o salven las vidas de individuos y familiares.

Poder conocer las mutaciones en los genes de cada individuo permite un mejor diagnóstico, y que se pueda ofrecer una mayor calidad en el asesoramiento a los pacientes y las familias respecto a la afección que padecen o tengan el riesgo de padecer. También puede ayudar a la toma de decisiones importantes de cara a la futura descendencia (enfermedades hereditarias).

Teniendo información sobre las características genéticas de un individuo, se pueden establecer tratamientos médicos personalizados adecuados, consiguiendo así una reducción en el gasto económico en fármacos, tanto en su utilización como en su desarrollo. Al mismo tiempo, se pueden minimizar los efectos secundarios que pueda sufrir el paciente debido a la toma de los mismos, al no tener que probar una determinada cantidad de ellos hasta encontrar el más eficaz.

Por estos motivos, se hace necesario el desarrollo e implementación de métodos y herramientas que permitan el acceso, procesado y manejo de diferentes tipos de información.

Las tecnologías de la información juegan un papel fundamental. La utilización de los conocimientos en genética y las nuevas tecnologías son necesarios para el mantenimiento del sistema sanitario, no sólo a nivel paliativo sino también preventivo, ya que la identificación de las causas genéticas de las enfermedades permitirán el desarrollo de mejores métodos

de diagnóstico y de fármacos personalizados adecuados a la afección de cada paciente.

La información obtenida de las técnicas y herramientas *bioinformáticas*[2] son de gran utilidad para ayudar a interpretar los resultados cosechados del estudio del genoma humano. Cuanto mayor sea la confianza que aporte esta información, menor será el gasto económico de laboratorio necesario para comprobar la veracidad de los datos obtenidos.

De la misma forma, se puede conseguir una disminución de la inversión en tiempo para el análisis de los distintos resultados en función de la calidad, cantidad y organización de la información resultante.

4. DESARROLLO

4.1 METODOLOGÍA DE TRABAJO

Para la implementación de las aplicaciones seguiremos un proceso de desarrollo por etapas en el que se llevarán a cabo las tareas necesarias para satisfacer cada uno de los objetivos. En este proceso el cliente tendrá una participación activa. En nuestro caso, el cliente será el Jefe de Servicio de la UICHUIMI.

Como un primer paso, se harán reuniones con el cliente para exponer el problema a resolver dependiendo del objetivo que se tenga entre manos en cada etapa, así como las distintas particularidades que presenten cada uno de ellos. Posteriormente se planteará una o varias soluciones que serán nuevamente expuestas y discutidas con el cliente para elegir la que se considere mejor.

Una vez elegida la opción que se estime más adecuada para solucionar el problema, se implementará la herramienta que le de respuesta. Cuando la aplicación sea funcional, se mostrará nuevamente al cliente con el fin de poder hacer las distintas pruebas que se consideren oportunas de cara a validar la utilidad o bien hacer las correcciones pertinentes.

Cuando se haya dado por válida la aplicación, se pasará al siguiente objetivo y se repetirá el proceso hasta haberlos cubierto todos.

4.2 RECURSOS NECESARIOS

Para poder implementar las aplicaciones necesitaremos una serie de materiales que listamos a continuación:

- Ordenador de sobremesa: utilizaremos el equipo proporcionado por la UICHUIMI, ya que debido al tamaño y la cantidad de información de los ficheros que utilizaremos, necesitamos una computadora con características que nos permitan trabajar con ellos de forma eficiente. Este equipo tiene instalado como sistema operativo una distribución de *Linux*. Inicialmente se empleó *CentOS* y posteriormente fue sustituido por *Debian*.

- Lenguaje de programación *Java*: se ha decidido utilizar este lenguaje porque es el mismo que se empleó para desarrollar la aplicación global a la que se agregarán las utilidades elaboradas en este TFT. De esta forma facilitaremos la integración de las mismas.

- Entorno de desarrollo: emplearemos *NetBeans* como entorno de desarrollo.

- Herramientas relacionadas con el análisis del ADN: con propósitos puramente ilustrativos, y para poder entender e interpretar la información de distintos tipos de ficheros que contienen información genética, se han utilizado algunas herramientas, como por ejemplo *SAMtools* [12] o *Integrative Genomics Viewer (IGV)* [21].

4.3 COMBINATOR

Combinator es una herramienta desarrollada con el fin de dar respuesta a las tres primeras metas que queremos alcanzar:

1. Combinar variantes de varias muestras.

1.1 Buscar variantes comunes en los pacientes seleccionados.

1.2 Excluir variantes de personas no afectadas (controles).

2. Incorporar la funcionalidad de identificar variantes que se encuentran en una zona de baja cobertura (MIST).

3. Anotar las variantes resultantes (añadir la información obtenida de uno o varios anotadores).

4.3.1 COMBINAR VARIANTES DE VARIAS MUESTRAS

La finalidad de esta utilidad es poder combinar los datos de interés de diferentes ficheros en formato *Variant Call Format (VCF)* [12] (ver anexo 8.1) teniendo en cuenta una serie de condiciones.

Se hará un filtrado entre dos grupos de ficheros de entrada (*pacientes* y *controles*), buscando aquellas líneas de los ficheros de pacientes en las que coinciden el cromosoma (*chrom*) y la posición (*pos*) para todos y cada uno de ellos. Pero además, hemos de tener en cuenta que esas posiciones no se encuentren en ninguno de los ficheros de control.

En la tabla 4.1 vemos las condiciones que se tienen que dar para que un grupo de líneas sea *válido* o *no válido*.

Localización (chrom:pos)	Paciente 1	Paciente 2	Control 1	Control 2	Resultado
1:10567	X	X	-	-	Válido
3:103478	X	X	-	X	No válido
18:135723	X	-	X	-	No válido

Tabla 4.1: e*jemplo de distintas situaciones que podemos encontrarnos a la hora de buscar coincidencias entre ficheros de pacientes y controles. Si la posición especificada se encuentra en el archivo se indica con "X" y en caso contrario se indica con "-".*

Tras el filtrado, se almacenarán en un archivo de salida las líneas pertenecientes al fichero del paciente 1 para las posiciones en las que se cumplan las condiciones anteriormente descritas. El fichero paciente 1 será el que utilizaremos como *referencia a la hora de hacer las comparaciones*.

De la información contenida en el fichero de referencia, nos quedaremos con aquellos campos que sean de interés para el usuario, descartando información que no se estime necesaria. En nuestro caso, se modificará el campo *info* de forma que tomemos los subcampos que necesita el usuario, como pueden ser la frecuencia de los alelos o el número de lecturas para las muestras. El resto de campos se replican del fichero de referencia.

Veamos un ejemplo de búsqueda de posiciones coincidentes:

Localización (chrom:pos)	Variante (Ref/Alt)	Paciente 1	Paciente 2	Control 1	Control 2	Resultado
1:753541	G/A	X	-	-	-	No válido
1:808631	G/A	X	X	-	-	Válido
1:808922	G/A	X	X	X	X	No válido
1:808928	C/T	X	X	X	X	No válido
1:856329	C/G	X	-	-	-	No válido

Tabla 4.2: *ejemplo de búsqueda de posiciones coincidentes para cuatro archivos de entrada (dos de pacientes y dos controles). El símbolo "X" indica que se ha encontrado la posición mientras que el símbolo "-" indica que no se ha encontrado.*

En el ejemplo de la tabla 4.2 se ha encontrado que en la posición 808631 del cromosoma 1 tenemos una variante que aparece en todos los ficheros de pacientes pero no en los controles. Por lo tanto, al cumplir con nuestras condiciones de búsqueda, la línea del fichero de referencia correspondiente a esta posición acabará volcada en el fichero de salida. A continuación tenemos un ejemplo de la línea que figurará en el archivo de salida:

```
#CHROM  POS     REF     ALT     INFO
1       808613  G       A       AF=0.500;DP=18
```

Código 4.1: *ejemplo de línea del fichero de salida en la que vemos los campos con la información que desea el usuario.*

Podemos observar que se indica la variante (*chrom* y *pos*), los *alelos de referencia y alterno* (*ref* y *alt* respectivamente), y la información que el usuario desea conservar, en este caso, la *frecuencia de alelos* (*AF*) y el *número de lecturas para cada muestra* (*DP*).

En el anexo 8.2 podemos ver un ejemplo del fichero de salida obtenido.

4.3.1.1 IMPLEMENTACIÓN

El proceso para hacer las comparaciones consistirá en tomar como referencia el primer archivo para recorrerlo por orden y, por cada variante, buscamos en los demás si la contienen.

Almacenamos en un buffer (en nuestro caso utilizaremos un vector) la última línea (o variante) leída en cada fichero, como vemos en la figura 4.1.

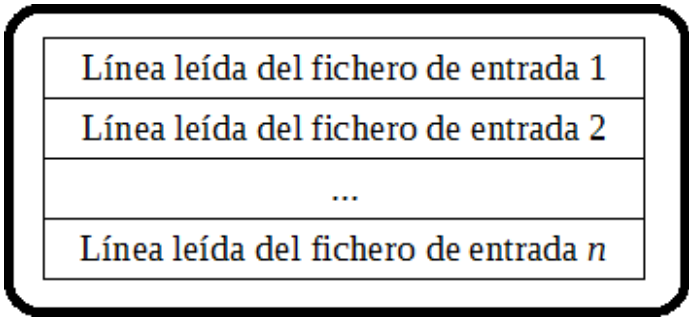

Figura 4.1: *buffer en el que se almacenan las líneas leídas de los ficheros de entrada en cada iteración.*

En cada fichero se avanzará hasta llegar a la posición de referencia, si se encuentra, o hasta la primera posición que sea mayor.

Una vez que se ha comprobado si la posición actual se encuentra o no en cada uno de los archivos de entrada, se pasa a la siguiente línea del fichero de referencia y se repite el proceso para este nuevo par de cromosoma y posición.

Es importante destacar que en un fichero vcf los cromosomas se encuentran ordenados del cromosoma 1 al cromosoma 22, seguidos del cromosoma X, cromosoma Y, y el cromosoma MT. De la misma manera, las posiciones dentro de cada cromosoma se encuentran ordenadas de menor a mayor.

A la hora de comparar los cromosomas, lo haremos de forma numérica. Para ello, almacenaremos los cromosomas en un vector de manera que su posición dentro del mismo se corresponderá con su valor numérico, como podemos observar en la siguiente tabla:

Cromosoma	Valor	Cromosoma	Valor	Cromosoma	Valor
1	0	*10*	9	*19*	18
2	1	*11*	10	*20*	19
3	2	*12*	11	*21*	20
4	3	*13*	12	*22*	21
5	4	*14*	13	*X*	22
6	5	*15*	14	*Y*	23
7	6	*16*	15	*MT*	24
8	7	*17*	16		
9	8	*18*	17		

Tabla 4.3: *valor numérico de los cromosomas para hacer las comparaciones.*

Para llevar a cabo las búsquedas de posiciones coincidentes entre dos líneas leídas, siempre se comparará primero el cromosoma, y si son iguales, pasaremos a comprobar la posición dentro de ese cromosoma.

Al comparar los cromosomas, si se da el caso en el que sean distintos, hemos de comprobar cuál es mayor y cuál es menor, respecto al orden, para saber si hemos de avanzar en la lectura del fichero de referencia, o si por el contrario hemos de avanzar en la lectura del archivo a comparar.

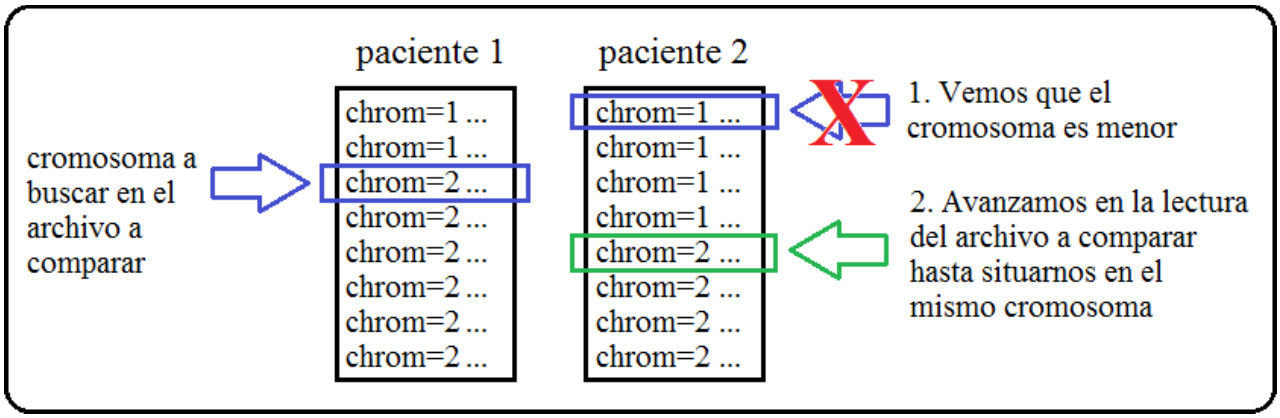

Figura 4.2: *ejemplo del caso en el que hemos de avanzar en la lectura del fichero a comparar porque el cromosoma en el archivo de referencia es mayor.*

Para comparar las posiciones utilizaremos una estrategia similar a la empleada para los cromosomas.

Empleando esta estrategia de comparación conseguimos evitar tener que leer desde el principio el fichero a comparar por cada línea del fichero de referencia.

En el código 4.2 se muestra con pseudocódigo el algoritmo utilizado para la combinación de variantes.

```
Para cada variante en la referencia:
      para cada archivo:
           para cada variante en archivo:
                si variante en referencia = variante en archivo:
                     variante encontrada
                 si variante en referencia < variante en archivo:
                     avanzar en la referencia
                     volver al primer archivo
```

Código 4.2: *algoritmo utilizado para la búsqueda de variantes comunes.*

Hemos de tener en cuenta que un fichero vcf contiene cientos de miles de líneas de información, y que la herramienta está pensada para trabajar con varios ficheros de este tipo.

En el peor de los casos (última posición del cromosoma MT) habría que recorrer completos cada fichero a comparar para cada línea del fichero de referencia. De esta manera, iremos quedándonos con la última posición que hemos leído de cada uno de los ficheros de entrada sin tener que retornar al comienzo de estos archivos, como si de un marcador se tratase.

Una vez que nos encontramos en condiciones para comparar cromosoma y posición entre las líneas leídas de los ficheros de entrada, tendremos que verificar que las coincidencias encontradas se dan en todos y

cada uno de los archivos de los distintos pacientes, pero no en los controles. Esta situación la veíamos descrita en la tabla 4.1.

Con el fin de comprobar las coincidencias encontradas, nos apoyaremos en un vector que tendrá tantos elementos como ficheros de entrada.

La utilización de este vector consistirá en que almacenaremos en él las líneas de los archivos de entrada para las que se hayan encontrado coincidencias respecto a cromosoma y posición. Desde el momento en que encontremos una coincidencia, se guardarán en el vector la línea del archivo de referencia (paciente 1) y la línea del fichero con el que se produjo la coincidencia, siendo *nulo (null)* el valor del elemento del vector correspondiente a un fichero para el que no se haya encontrado una posición coincidente. En la tabla 4.4 vemos un ejemplo.

Pacientes	**¿Existe la variante?**	**Resultado (valor en el vector)**
Paciente 1	X	Línea leída paciente 1
Paciente 2	X	Línea leída paciente 2
Paciente 3	X	Línea leída paciente 3
Control 1	-	*nulo*
Control 2	-	*nulo*

Tabla 4.4: e*jemplo de uso del vector para comprobar las coincidencias que ilustra la situación en la que se encuentran coincidencias para todos los archivos de pacientes pero no en los controles. si se ha encontrado una coincidencia con el archivo de referencia (existe la variante) se indica con "X" y en caso contrario se indica con "-".*

A continuación se comprobarán todos los elementos del vector para ver si las coincidencias correspondientes a pacientes son distintas de *nulo*, mientras que por otro lado, los elementos del vector correspondientes a controles deberán tener el valor *nulo*. Si se cumplen estas condiciones, tendremos una de las posiciones que buscamos (variante común encontrada).

Por otra parte, para todas las posiciones cuyas líneas acabarán volcadas en el fichero de salida, tomaremos la línea original del fichero de referencia y a partir de ella se generará una nueva línea en la que figuren los campos que el usuario desea (como vimos en el apartado 4.3.1).

Internamente, tendremos un vector en el que aparece el listado de los campos que el usuario quiere, y que tomaremos de la línea leída para poder construir la línea que aparecerá en el fichero de salida. En el código 4.1 tenemos un ejemplo de línea de salida.

Por último, queremos que el fichero de salida mantenga el formato vcf, y por lo tanto, tendremos que añadir las correspondientes líneas de cabecera.

Para llevar a cabo esta tarea, tomaremos las líneas de cabecera del fichero de referencia (paciente 1), pero descartaremos aquellas cabeceras correspondientes a los campos que no figurarán en el archivo de salida.

Por otro lado, además de las líneas de cabecera que hemos replicado del vcf de referencia, añadiremos otras nuevas para especificar los ficheros de entrada correspondientes a pacientes (denominados ***include***) y los controles empleados (denominados ***exclude***). Podemos ver ejemplos de ellas a continuación:

```
##Reference_Include=File:/home/Investigacion/Archivos_pacientes/
paciente1.vcf
##Include=File:/home/Investigacion/Archivos_pacientes/paciente2.
vcf
##Exclude=File:/home/Investigacion/Archivos_control/control1.vcf
##Exclude=File:/home/Investigacion/Archivos_control/control2.vcf
```

Código 4.3: *ejemplos de líneas de cabecera que indican la ubicación de los ficheros de pacientes (include) y los de control (exclude) utilizados en una ejecución concreta de la herramienta. Además, se especifica el archivo del paciente que utilizamos como referencia (Reference_Include).*

4.3.2 IDENTIFICAR VARIANTES QUE SE ENCUENTRAN EN UNA ZONA DE BAJA COBERTURA

El objetivo es añadir a ***Combinator*** una funcionalidad que nos permita localizar aquellas posiciones de los archivos de entrada que se encuentren dentro de una zona de baja cobertura (ver apartado 1.4). A estas regiones las llamaremos *zonas mist* (por la herramienta *MIST* [19]) y las obtendremos como la intersección entre un exón y una región pobre o de baja lectura.

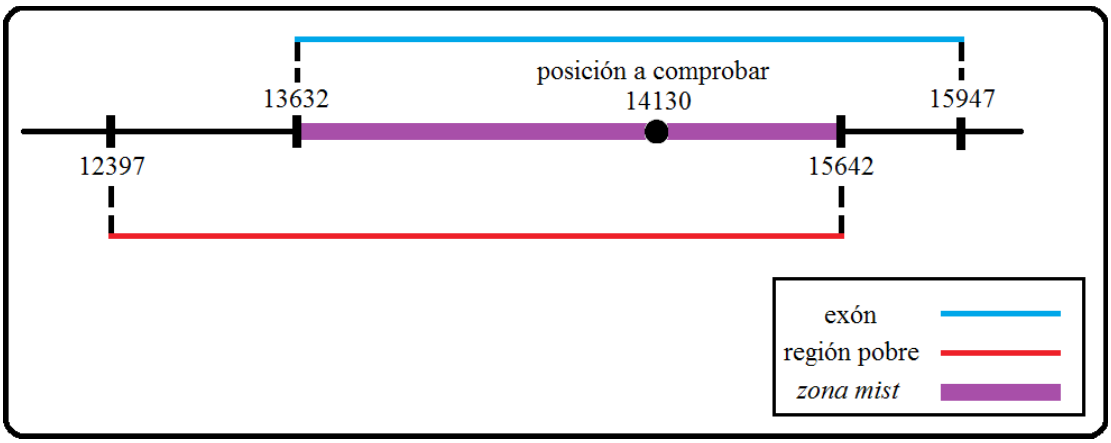

Figura 4.3: *ejemplo en el que vemos que una posición que deseamos comprobar se encuentra dentro de una zona mist (intersección de exón y región pobre).*

Para ello, necesitaremos como entrada una pareja de ficheros por cada paciente que deseemos analizar, uno en formato vcf (ver anexo 8.1) y el otro en formato *mist* (ver anexo 8.3), por ejemplo: *paciente1.vcf* y *paciente1.mist.*

El fichero en formato mist lo obtenemos de la herramienta MIST (ver apartado 1.4), y en él se especifican una serie de exones y regiones pobres de las que obtendremos las zonas mist para el paciente al que está asociado.

Al encontrar una posición que esté en una zona mist, añadiremos una nueva etiqueta denominada ***MistZone*** como subcampo del campo *info*, en la

línea correspondiente del fichero de salida (en formato vcf) para indicar tal situación. El fichero de salida será idéntico al de entrada, pero aparecerá la nueva etiqueta ***MistZone*** en las líneas que corresponda.

```
#CHROM    POS         REF   ALT   INFO
1         22902523    C     T
1         22915967    C     G     MistZone
1         22922546    G     A
1         22923873    G     C
1         23405872    T     C
1         23664797    A     G     MistZone
1         23744382    A     G
1         23744691    C     T     MistZone
```

Código 4.4: *ejemplo de líneas del fichero de salida en el que vemos que se ha añadido la etiqueta MistZone en las posiciones que se encuentran en una intersección de exón y región pobre.*

En el anexo 8.4 podemos ver un ejemplo del fichero de salida obtenido.

Cabe destacar que esta funcionalidad podemos utilizarla de dos formas diferentes:

Por un lado, podemos emplearla dentro de la utilidad para combinar (ver apartado 4.3.1), de forma que indica si las variantes coincidentes encontradas para los distintos pacientes se localizan o no en una zona mist. En este caso, la combinación funciona de la siguiente manera:

- Una variante es coincidente si todos los pacientes la tienen y los controles no.
- Una variantes no es coincidente si al menos un paciente no la tiene o si al menos un control la tiene.
- Una variante es indeterminada (hay una zona mist) si todos los pacientes tienen la variante o son indeterminados, o si todos los controles no tienen la variante o son indeterminados.

Por otro lado, podemos utilizar la funcionalidad de forma independiente, pasándole uno o varios ficheros de pacientes para que nos indique las variantes de cada uno de ellos que se encuentran en zonas mist.

4.3.2.1 IMPLEMENTACIÓN

Antes de explicar con mayor detalle el funcionamiento de esta nueva funcionalidad, hemos de tener en cuenta que para el desarrollo de la misma se ha partido de la base de que *el fichero vcf y el mist correspondientes a un determinado individuo se encuentran en el mismo directorio*.

Figura 4.4: *carpeta en la que tenemos una serie de ficheros vcf y varios ficheros mist correspondientes a algunos de ellos.*

A partir del fichero vcf, obtenemos la ruta y el nombre del fichero mist correspondiente. Es condición necesaria que ambos archivos tengan el mismo nombre, pero cada uno con su extensión pertinente (.vcf y .mist).

El proceso de detección de posiciones que se encuentran en zonas mist lo hemos dividido en dos fases:

1. Obtener las zonas mist contenidas en el fichero mist.

2. Identificar las posiciones del archivo vcf que se encuentran en una zona mist.

A continuación vemos detalladas ambas fases.

1. Obtener las zonas mist contenidas en el fichero mist.

Un fichero mist tiene una serie de campos, de los que en nuestro caso solamente necesitaremos los cinco primeros, que vemos descritos a continuación:

- *chrom:* cromosoma. En este tipo de ficheros aparecen las zonas mist correspondientes a los cromosomas del 1 al 22, X e Y.
- *exon_start:* posición de comienzo de la región del exón.
- *exon_end:* posición que indica el final de la región del exón.
- *poor_start:* posición de comienzo de la región pobre.
- *poor_end:* posición que indica el final de la región pobre.

A partir de estos cinco campos, obtendremos tres nuevos campos con los que definiremos una zona mist:

- *chrom_mist_zone:* cromosoma. Corresponde al mismo valor que el campo *chrom* del archivo mist.
- *start:* inicio de intersección de exón y región pobre. De los campos *exon_start* y *poor_start* del fichero mist, tomamos el de mayor valor.
- *end:* final de intersección de exón y región pobre. De los campos *exon_end* y *poor_end* que tenemos en el fichero mist, nos quedaremos con el de menor valor.

Veamos la obtención de estos tres campos con un ejemplo. Supongamos que tenemos la siguiente línea extraída de un fichero mist:

```
chrom    exon_start    exon_end    poor_start    poor_end
1         897206        897858      897445        897660
```

Código 4.5: *ejemplo de línea leída de un fichero mist en el que tenemos los cinco campos que nos interesan para definir una zona mist.*

Campo **chrom_mist_zone = 1**.

En este caso, **(poor_start = 897445)** > **(exon_start = 897206)**, por lo tanto, **start = 897445**.

Por último, **(poor_end = 897660)** < **(exon_end = 897858)**, por tanto, **end=897660**.

Finalmente, la zona mist obtenida será la definida por:

```
chrom_mist_zone    start      end
1                   897445     897660
```

Código 4.6: *ejemplo de zona mist obtenida a partir de la información de un archivo mist.*

A continuación, y a modo de resumen, mostramos el algoritmo utilizado para la obtención de zonas mist a partir del fichero mist:

```
Para cada línea en archivo:
      cromosoma → cromosoma_archivo
      inicio → máximo(inicio_exón, inicio_mist)
      fin → mínimo(fin_exón, fin_mist)
```

Código 4.7: *algoritmo de obtención de zonas mist a partir de la información leída de un archivo mist.*

Posteriormente, almacenaremos las zonas mist que hemos obtenido en una estructura de tipo *Map*, donde la clave es el cromosoma (*chrom_mist_zone*). Cada clave tiene asociada una lista cuyos elementos corresponden a las distintas zonas mist (*start, end*) para ese cromosoma, como vemos en la siguiente figura:

Cromosoma (*chrom_mist_zone*)	Zonas mist (*start, end*)
1	start =… end =…
	start =… end =…
	...
2	start =… end =…
	...
...	start =… end =…
	...
X	start =… end =…
	...
Y	start =… end =…
	...

Figura 4.5: *ejemplo que ilustra la estructura Map en la que se almacenan las diferentes zonas mist extraídas del archivo mist. Tenemos como clave el cromosoma y una lista de zonas mist asociadas a cada uno.*

2. Identificar las posiciones del archivo vcf que se encuentran en una zona mist.

Una vez que tenemos las zonas mist almacenadas en el Map, el objetivo será recorrer el fichero vcf de entrada y comprobar si las posiciones que tenemos se encuentran dentro de alguna de las intersecciones de exón y región pobre que hemos obtenido del archivo mist.

La comprobación se hará de forma sencilla. Primero comparamos el cromosoma leído de la línea del fichero vcf con el cromosoma de la zona mist (*chrom_mist_zone*). Si los cromosomas coinciden, la segunda comprobación consistirá en ver si la posición leída del vcf se encuentra dentro del rango que definen los campos *start* y *end*, en alguna de las zonas mist que tengamos para ese cromosoma. En caso afirmativo, tendremos una posición que se encuentra en una intersección de exón y región pobre.

```
Para cada línea en el vcf:
      Para cada zona mist del map:
          si cromosoma_vcf = cromosoma_map:
              si posición_vcf >= inicio_zona_mist
              y posición_vcf <= fin_zona_mist:
                  la variante está en una zona mist
```

Código 4.8: *algoritmo para identificar si una posición se encuentra dentro de una zona mist.*

Una vez que podemos comprobar cuántas posiciones de un fichero vcf se encuentran en una zona mist, generamos un archivo de salida en el que aparezca como subcampo del campo *info* la etiqueta a la que hemos llamado ***MistZone***, para todas aquellas posiciones situadas en una intersección de exón y región pobre. En el código 4.4 podemos ver algunas líneas de ejemplo para un fichero de salida obtenido de esta herramienta.

Por otro lado, como queremos que el archivo de salida mantenga el formato vcf, replicaremos las líneas de cabecera del fichero de entrada. Además, como la etiqueta ***MistZone*** será agregada como subcampo del campo *info*, tendremos que definir su correspondiente línea de cabecera.

```
##INFO=<ID=MistZone,Number=0,Type=Flag,Description="If
present, indicates that the position is in an Mist Zone">
```

Código 4.9: *línea de cabecera explicativa para la etiqueta MistZone.*

4.3.2.2 MEJORAS DE RENDIMIENTO

Antes de la utilización del Map como estructura para almacenar las zonas mist, se probó a utilizar una estructura de tipo *ArrayList* (ver figura 4.6) en la que se almacenaba el conjunto de zonas mist extraídas del fichero mist. Posteriormente se comparaba cada línea leída del fichero vcf con los elementos de esta lista para comprobar si estaban localizados dentro de una intersección de exón y región pobre.

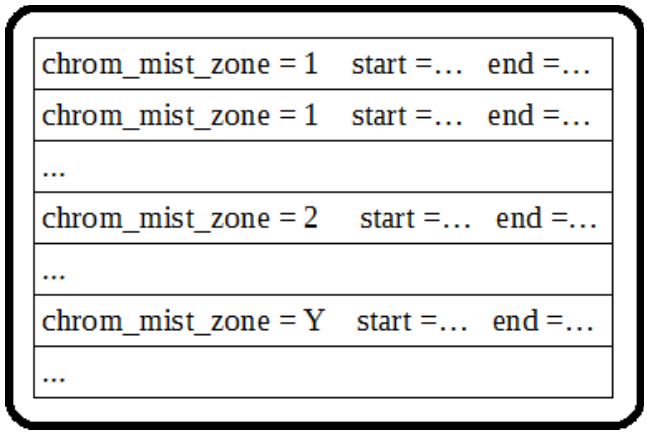

Figura 4.6: *ejemplo que ilustra la estructura ArrayList en la que se almacenan las diferentes zonas mist extraídas del archivo mist.*

El problema de esta solución es que para una pareja de ficheros vcf y mist, llegamos a encontrarnos con tiempos de ejecución de **más de 90 minutos**, por lo que se decidió descartar el uso de este tipo de estructura como almacén para las zonas mist.

La causa de un tiempo de ejecución tan elevado la encontramos en que, para cada cromosoma, era necesario recorrer toda la lista de zonas mist hasta llegar al cromosoma de interés, es decir, al avanzar en los cromosomas cada vez se tenía que recorrer más elementos de la lista.

En los peores casos (las posiciones del cromosoma Y) nos veíamos obligados a pasar por todas las zonas mist correspondientes a los cromosomas del 1 al 22 y el cromosoma X para cada una de las líneas leídas del fichero vcf correspondientes al cromosoma Y. Recordemos que este tipo de ficheros pueden contener cientos de miles de líneas de información.

Para mitigar ese inconveniente, se probó a utilizar una estructura de tipo Map, en la que empleamos como clave el cromosoma (*chrom_mist_zone*), teniendo cada clave una lista asociada cuyos elementos corresponden a las distintas zona mist para ese cromosoma (*start, end*). Esta estructura Map la veíamos ilustrada en la figura 4.5.

De esta manera, accedemos directamente al cromosoma que nos interese en cada caso para comprobar si la posición se encuentra en alguna de las zonas mist que tenemos en la lista.

Con esta modificación conseguimos que el tiempo de ejecución empleando la misma pareja de ficheros vcf y mist se redujese aproximadamente hasta **1 minuto y 40 segundos**.

Además, después de realizar varias pruebas de funcionamiento, nos dimos cuenta de que en muchos casos las zonas mist obtenidas podían encontrarse desordenadas e incluso haber intersecciones de exón y región pobre repetidas.

Por lo tanto, y sabiendo que las posiciones de un fichero vcf se encuentran ordenadas de menor a mayor, se optó por ordenar las zonas mist de menor a mayor, y eliminar las zonas repetidas. De esta forma, avanzamos un paso más en la reducción del tiempo de ejecución de la herramienta.

La ordenación de los elementos se hará en base al campo *start*, y si este campo es igual en los dos elementos que queremos ordenar, se hará respecto al campo *end*. Si nos encontramos con dos intersecciones iguales, no se insertará la segunda en el Map para evitar tener elementos repetidos.

Zonas mist (antes de la ordenación)	**Zonas mist (después de la ordenación)**
(11869, 12139)	(11869, 12139)
(12145, 12146)	(11872, 12139)
(11872, 12139)	(11874, 12139)
(12145, 12146)	(12145, 12146)
(11874, 12139)	

Tabla 4.5: *lista de zonas mist (start, end) antes y después de ordenar sus elementos. También se han eliminado las zonas mist repetidas.*

Una vez implementados el uso del Map y la ordenación de zonas mist, se probó nuevamente a buscar posiciones de un fichero vcf para comprobar cuántas se encontraban en una intersección de exón y región pobre. A continuación podemos ver un promedio de tiempos de ejecución obtenidos en diferentes pruebas:

N.º de parejas vcf y mist	Tiempo para lista de zonas mist sin ordenar	Tiempo para lista de zonas mist ordenada
1	3 minutos (aprox.)	2 minutos (aprox.)
6	10 minutos (aprox.)	5 minutos (aprox.)

Tabla 4.6: *tiempos de ejecución obtenidos para una serie de pruebas realizadas en las que comprobamos cuántas posiciones de un fichero vcf se encuentran en una zona mist.*

Podemos observar que al aumentar el número de parejas de ficheros de entrada la reducción de tiempo se hace considerable.

Esto resulta de gran interés ya que la herramienta está pensada para trabajar con varias parejas de vcf y mist, y recordemos que ambos tipos de ficheros pueden contener cientos de miles de líneas con información.

4.3.3 ANOTAR LAS VARIANTES RESULTANTES

Como última funcionalidad para ***Combinator***, queremos tomar un fichero vcf (ver anexo 8.1), al que llamaremos fichero base, y agregarle información adicional que nos aportan diferentes herramientas anotadoras.

En nuestro caso, utilizaremos los anotadores *Variant Effect Predictor (VEP)* [16], *Sort Intolerant From Tolerant (SIFT)* [17] y *Annotate Variation (ANNOVAR)* [18].

Como resultado, deseamos generar un archivo vcf idéntico al fichero base, pero que contenga las anotaciones que nos aportan los anotadores como subcampos del campo *info*.

Para ello partiremos del fichero base y el proporcionado por el anotador, que obtenemos al pasar el fichero base por el anotador que deseemos. En los anexos 8.5, 8.6 y 8.7 podemos ver ejemplos de los ficheros obtenidos de VEP, SIFT y ANNOVAR, respectivamente.

Utilizando estos ficheros como entrada para la herramienta conseguiremos un archivo de salida con un formato similar al que vemos a continuación:

```
#CHROM    POS         REF    ALT    INFO
1         22902523    C      T      Campo_Anotador=valor;...
1         22915967    C      G      Campo_Anotador=valor;...
1         22922546    G      A      Campo_Anotador=valor;...
1         22923873    G      C      Campo_Anotador=valor;...
1         22924464    T      C      Campo_Anotador=valor;...
1         23405872    T      C      Campo_Anotador=valor;...
1         23664797    A      G      Campo_Anotador=valor;...
```

Código 4.10: *ejemplo de la estructura que puede tener un fichero de salida una vez que se ha añadido la información proporcionada por la/s herramienta/s anotadora/s.*

En el anexo 8.8 podemos ver un ejemplo del fichero de salida obtenido tras añadir la información adicional del anotador VEP.

4.3.3.1 IMPLEMENTACIÓN

El primer paso será obtener el fichero que nos proporciona el anotador para el que deseemos obtener la información adicional. Dependiendo del anotador, podemos disponer de la funcionalidad de anotación de manera online o habrá que descargar la herramienta, instalarla y utilizarla de forma local en nuestro equipo.

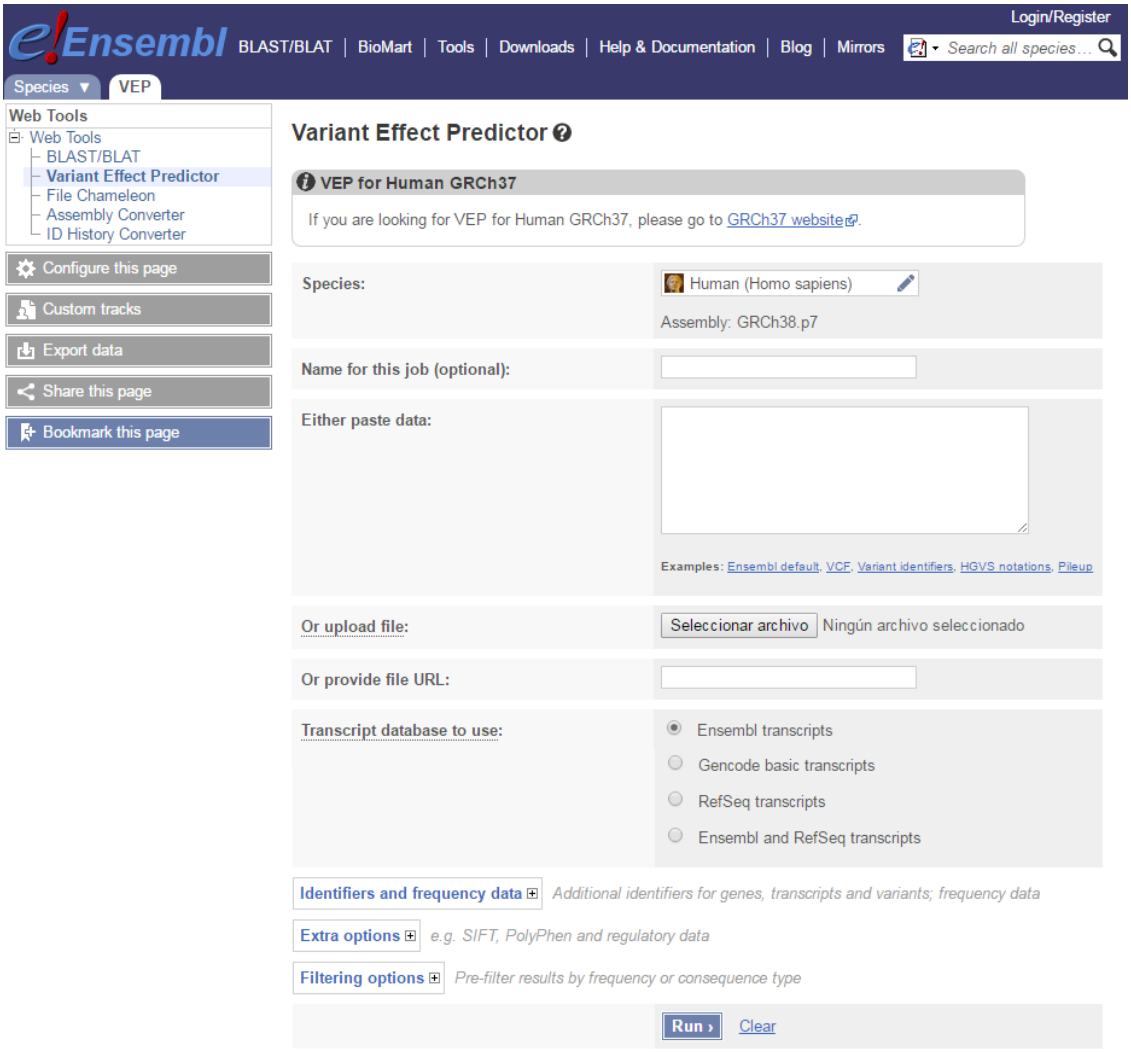

Figura 4.7: *herramienta online del anotador VEP. Vemos varias maneras para subir nuestro archivo vcf de entrada (paste data, upload file, provide file URL) así como diversas opciones que se pueden configurar según se necesite.*

Cada anotador tiene una serie de opciones que podemos modificar según lo que nos interese en cada caso.

Como resultado, tendremos un archivo en un formato determinado que depende de cada anotador (ver tabla 4.7), en el que figurará la información adicional añadida por la utilidad para cada una de las posiciones del fichero base, mediante una serie de campos.

Anotador	Formato
VEP	Fichero *txt* en el que los campos se encuentran separados por un tabulador.
SIFT	Fichero *tsv* en el que los campos se encuentran separados por un tabulador.
ANNOVAR	Fichero *csv* en el que los campos se encuentran separados por ‘**,**’ (coma).

Tabla 4.7: *formato en el que se encuentran los ficheros obtenidos de cada herramienta anotadora.*

Posteriormente, teniendo el fichero base y el obtenido de la herramienta anotadora, generamos un fichero de salida en formato vcf en el que figurará la información del fichero base y como subcampos del campo *info*, los campos que el usuario considere de interés de los obtenidos de los anotadores.

Internamente tendremos un vector en el que, al ir leyendo el fichero obtenido del anotador, almacenamos los distintos campos que el usuario necesite. Posteriormente, con la línea leída del fichero base y los campos almacenados en el vector, generamos cada una de las líneas que se escribirán en el archivo de salida.

Evidentemente, como queremos que la salida mantenga el formato vcf [12], debemos añadir líneas de cabecera explicativas para cada uno de los campos obtenidos del anotador que agreguemos a nuestro archivo.

```
##INFO=<ID=”Nombre del campo”,Number=”Número de
valores que pueden ser incluidos en el campo
INFO”,Type=”Tipo del campo”,Description=”Breve
descripción de la información que aporta el campo”>
```

Código 4.11: *ejemplo de formato de las líneas de cabecera para los subcampos de info.*

4.3.4 USO DE COMBINATOR

Como hemos visto en los apartados anteriores, la herramienta ***Combinator*** puede ser utilizada de tres formas básicas dependiendo de la funcionalidad que nos interese:

A) Combinar variantes de varias muestras.

B) Identificar variantes que se encuentran en zonas mist.

C) Anotar variantes.

A continuación describiremos los pasos a seguir para utilizar cada una de las funcionalidades.

A) Combinar variantes de varias muestras.

1. Especificar uno o varios archivos (en formato vcf) de pacientes para los que queramos combinar.
2. Especificar uno o varios archivos (en formato vcf) que queramos que hagan la labor de control. Este paso es opcional ya que nos puede interesar simplemente combinar las variantes de dos pacientes sin utilizar ningún control.
3. Ejecutar Combinator.
4. Como resultado, obtendremos un fichero similar al que podemos ver en el anexo 8.2.

B) Identificar variantes que se encuentran en una zona mist.

1. Especificar uno o varios archivos (en formato vcf).

Como vimos en el apartado 4.3.2.1, a partir del fichero vcf obtenemos la ruta y el nombre del fichero mist correspondiente, y por tanto, es condición

necesaria que exista un archivo mist con el mismo nombre que el vcf y en el mismo directorio.

2. Ejecutar Combinator.

3. Como resultado, tendremos un archivo similar al que podemos ver en el anexo 8.4, en el que se nos indican las variantes que están en una zona mist.

C) Anotar variantes.

1. Especificar un archivo vcf.

2. Especificar el archivo obtenido del anotador de nuestra elección.

   Para ello tomamos el fichero del paso 1 y lo pasamos por el anotador que nos interese (en nuestro caso: VEP, SIFT o ANNOVAR).

3. Ejecutar Combinator.

4. Como resultado, obtendremos un fichero similar al que podemos ver en el anexo 8.8, en el que se ha incluido la información del anotador elegido.

Cabe destacar que podremos añadir la información obtenida de varios anotadores. Por ejemplo, supongamos que deseamos agregar a un fichero vcf la información de los anotadores VEP y ANNOVAR. Para ello seguiremos los pasos descritos a continuación:

1. Pasar el fichero vcf (al que llamaremos base) por el anotador VEP para obtener el archivo con su información correspondiente.

2. Pasar el mismo fichero base por el anotador ANNOVAR para obtener el archivo con su información correspondiente.

3. Ejecutar Combinator empleando como ficheros de entrada el archivo base más el obtenido de VEP. Con esto conseguimos un fichero vcf al que se le ha agregado la información adicional de VEP.

4. Ejecutar Combinator utilizando como entradas el fichero obtenido en el paso 3 más el archivo de ANNOVAR del paso 2. Con esto conseguimos un fichero vcf en el que figura la información adicional proporcionada por VEP y ANNOVAR.

4.4 FALSE POSITIVE SEARCHER

La herramienta ***False Positive Searcher*** ha sido desarrollada con el fin de dar respuesta al cuarto objetivo de este TFT:

> **4.** Detectar las variantes que puedan ser *falsos positivos de tipo solapamiento* (ver apartado 1.5).

Dado un fichero vcf de entrada (ver anexo 8.1), analizaremos cada una de las variantes para detectar si pueden ser o no un falso positivo en base al método desarrollado para ello (ver apartado 4.4.1). Dicho método utilizará únicamente la información contenida en el vcf para hacer la identificación.

Finalmente, obtenemos un fichero de salida en formato vcf, en el que para aquellas líneas del fichero de entrada correspondientes a posiciones candidatas a ser falsos positivos, hemos añadido una nueva etiqueta como subcampo del campo *info*, denominada ***FP*** (*False Positive*), para indicar tal situación. Es decir, el archivo de salida será idéntico al de entrada con la salvedad de que se habrá añadido la etiqueta ***FP*** en las líneas pertinentes.

```
CHROM  POS          REF  ALT  INFO
7      147183121    A    C    FP
```

Código 4.12: *ejemplo de línea del fichero de salida en la que vemos que se ha añadido la etiqueta FP en una posición para la que False Positive Searcher ha determinado que se trata de un falso positivo potencial.*

En el anexo 8.9 podemos ver un ejemplo del fichero de salida obtenido.

4.4.1 MÉTODO DE BÚSQUEDA DE FALSOS POSITIVOS DE TIPO SOLAPAMIENTO

Para desarrollar un método que nos permita detectar los falsos positivos de tipo solapamiento, hemos partido del problema descrito en el apartado 1.5 y que, en lo referente al fichero vcf que utilizaremos como entrada, se traduce en que en la posición donde se produce el error de lectura (solapamiento) aparecerá una variante A/C (*alelo de referencia=A, alelo alterno=C*) cuando realmente no la hay.

```
CHROM   POS         REF   ALT
7       147183121   A     C
```

Código 4.13: *ejemplo de posición en un fichero vcf en el que vemos una variante producto de un falso positivo de tipo solapamiento.*

A la hora de localizar falsos positivos, hemos de tener en cuenta que la mayoría de ellos suelen encontrarse en variantes que en el archivo vcf figuran como heterocigotas [22, 23 y 24].

Por otro lado, algunos campos del fichero vcf que aportan probabilidades en *escala Phred* [25 y 26] para campos relacionados con el genotipo, pueden ser de gran utilidad para detectar falsos positivos [22 y 23]. Además, se deben tener en cuenta aquellas variantes con baja calidad.

En un primer paso, hemos recopilado una serie de posiciones de varios ficheros vcf para las que se tiene la certeza de que son falsos positivos de tipo solapamiento. En base a estas muestras, se buscarán puntos comunes que nos puedan ayudar a desarrollar un método de detección utilizando únicamente la información que tenemos disponible en los ficheros vcf, sin tener que recurrir a otra información de entrada.

Para la búsqueda de puntos comunes en las diferentes muestras, nos centraremos en una serie de campos del fichero vcf que están relacionados con el genotipo y las probabilidades asociadas a los mismos. Vemos dichos campos descritos a continuación:

- *GT* [12]*:* subcampo del campo *format* que nos indica el genotipo asignado a la variante (genotipo con mayor probabilidad de ser el correcto). En su forma más simple puede tener tres valores distintos:
  - *0/0:* indica que la variante es *homocigota referencia* u *homocigota silvestre* (homocigota para el alelo de referencia).
  - *0/1:* indica que la variante es *heterocigota*.
  - *1/1:* indica que la variante es *homocigota alterna (*homocigota para el alelo alterno).

- *PL* [12]*:* subcampo del campo *format*. Este campo contiene tres valores (separados por coma "**,**") en el siguiente formato:

  probabilidad 0/0, probabilidad 0/1, probabilidad 1/1

  Cada valor se corresponde con la probabilidad que tiene cada uno de los posibles valores del campo GT (0/0, 0/1 ó 1/1) de ser correcto, y se calculan como:

  $$PL = -10 * \log P\,(Genotype|Data)$$

  Donde *(Genotype|Data)* es la probabilidad condicional del genotipo en la muestra. Posteriormente, estos valores se normalizan de manera que uno de ellos será 0 (cero), que es la probabilidad correspondiente al genotipo que figura en el campo GT. Los otros dos valores están escalados respecto a este y serán números mayores que 0 (cero).

Cuanto menor es el valor, mayor probabilidad de que el genotipo sea el correcto, por lo tanto, si el segundo valor más pequeño es muy cercano a 0, no tendremos una total seguridad de que el genotipo correcto sea aquel cuyo valor es 0.

Por ejemplo, tenemos un campo PL con los valores: 103, 0, 18. Vemos que el genotipo 0/1 tiene asignado el valor de probabilidad 0 y el genotipo 1/1 tiene asignado el valor de probabilidad 18. Por tanto, al ser 18 es un valor muy cercano a 0, no podemos estar seguros de que el genotipo realmente sea 0/1 en lugar de 1/1.

- *GQ* [12]*:* subcampo del campo *format* que indica la probabilidad de que el genotipo asignado al campo GT sea correcto. Tendrá como valor máximo 99 y se calcula como la resta entre los dos valores menores de PL, es decir, a efectos prácticos será el segundo menor valor de PL, porque recordemos que el valor menor siempre es 0 (cero).

En el ejemplo anterior (PL=103, 0, 18), el valor para este campo sería GQ=18.

Hemos de tener en cuenta que si el segundo valor menor de PL es mayor que 99, será recortado a 99 porque es el máximo valor que permite este campo.

- *QD* [13]*:* subcampo del campo *info* que se calcula como ***QUAL/AD***. Nos aporta un valor de calidad normalizado al dividir la calidad entre el número de lecturas que dan soporte a los distintos alelos leídos. Este subcampo es dado por la herramienta GATK [13].

A continuación podemos ver algunas de las muestras extraídas de distintos vcf (para la misma posición) que analizaremos para tratar de encontrar puntos comunes que nos permitan detectar los falsos positivos:

SAMPLE	QD	GT	GQ	PL
1	2.46	0/1	99	139,0,478
2	1.39	0/1	88	88,0,464
3	1.49	0/1	91	91,0,413
4	1.97	0/1	99	121,0,495
5	0.73	0/1	61	61,0,457
6	3.52	0/1	99	215,0,507
7	0.82	0/1	61	61,0,466
8	0.65	0/1	59	59,0,713

Tabla 4.8: *muestras extraídas de una posición determinada de varios ficheros vcf para las que tenemos la certeza de que se tratan de falsos positivos de tipo solapamiento.*

En base al análisis de las muestras hemos llegado a las siguientes ***hipótesis***:

A) Las distintas variantes de las muestras tienen su campo GT = 0/1, es decir, lo más probable es que su genotipo sea heterocigoto. Como ya sabemos, las variantes heterocigotas son las más susceptibles de ser falsos positivos [22, 23 y 24].

B) La diferencia de los valores de PL es menor entre los correspondientes a los genotipos 0/0 y 0/1.

En algunos casos podría ser que la variante sea homocigota silvestre (0/0) y no heterocigota (0/1) como nos indica el campo GT. Con estas condiciones, es normal pensar que pueda tratarse de un falso positivo ya que tenemos la posibilidad de que no exista la variante.

Por otro lado, esta situación es algo que cabe esperar para un falso positivo potencial, puesto que si se diese el caso en el que la supuesta variante fuese heterocigota (0/1) y el segundo genotipo más probable fuese homocigoto alterno (1/1), no lo consideraríamos una situación "peligrosa" ya que en ambos casos existiría la variante.

C) En varios casos, el campo GQ tiene un valor menor que 99, lo que nos hace llegar a una hipótesis similar a B). Puede que el genotipo indicado en el campo GT no sea realmente el correcto para esa posición.

Al ser 99 el valor máximo que admite GQ, tomaremos como falsos positivos potenciales aquellas posiciones para las que GQ < 99.

D) Las calidades indicadas en el campo QD son bajas, lo que podría ser un indicio de que se trata de un falso positivo.

Teniendo en cuenta estas observaciones, hemos desarrollado la herramienta ***False Positive Searcher***. El proceso de identificación de falsos positivos se llevará a cabo de la siguiente manera:

1. Buscar aquellas variantes que sean heterocigotas, GT = 0/1 (ver hipótesis A).

2. Comprobar que el valor de PL para el genotipo 0/0 sea menor que el valor de PL para el genotipo 1/1. Buscamos aquellas variantes que figuren como heterocigotas y que como segundo genotipo más probable sean homocigotas silvestres (ver hipótesis B).

3. Comprobar valor de GQ.

3.1 Si el valor de GQ < 99, ***señalar la variante como falso positivo potencial*** (ver hipótesis C).

3.2. Si el valor de GQ >= 99, comprobar campo QD.

3.2.1 Si el valor de QD < umbral de calidad, ***señalar la variante como un posible falso positivo*** (ver hipótesis D). El valor para el umbral lo podremos establecer según la calidad que deseemos considerar como mínima aceptable.

A continuación podemos ver a modo de resumen el algoritmo utilizado para identificar los posibles falsos positivos de tipo solapamiento:

```
Para cada línea en el vcf:
      Si GT = 0/1:
          Si PL(0/0) < PL(1/1):
              Si GQ < 99:
                  Posible falso positivo
              Si no:
                  Si QD < umbral_de_calidad:
                      Posible falso positivo
```

Código 4.14: *algoritmo para detectar los posibles falsos positivos de tipo solapamiento en base a los campos GT, PL, GQ y QD, y a las conclusiones obtenidas de la búsqueda de puntos comunes de varias muestras.*

A todas aquellas posiciones para las que se haya determinado que pueden ser falsos positivos, les añadimos como subcampo del campo *info* una etiqueta denominada ***FP*** (ver código 4.12).

Además, como queremos que el archivo de salida conserve el formato vcf, debemos añadir la línea de cabecera descriptiva para la nueva etiqueta.

```
##INFO=<ID=FP,Number=0,Type=Flag,Description="If
present, indicates that the variant is likely a
false positive">
```

Código 4.15: *línea de cabecera explicativa para la etiqueta FP.*

4.4.2 PRUEBAS DE FUNCIONAMIENTO

Finalmente, comprobaremos si las variantes marcadas por ***False Positive Searcher*** (etiqueta ***FP***) son realmente falsos positivos.

Para realizar estas comprobaciones, utilizaremos varios ficheros vcf obtenidos de ***False Positive Searcher*** y un fichero *SNP Array*[19], que resulta de la detección directa, por un método de hibridación, de múltiples posiciones ya conocidas en el genoma que son susceptibles de presentar variaciones interindividuales.

En un archivo SNP Array (ver anexo 8.10) nos encontramos una serie de posiciones para muestras de varios pacientes para las que tenemos una alta fiabilidad de que los datos indicados para el genotipo sean correctos. Nos puede servir para comprobar si una variante especificada en este SNP Array se trata de un verdadero positivo o de un falso positivo.

Lo que haremos será una intersección entre nuestro fichero de salida de ***False Positive Searcher*** y el SNP Array, de manera que comprobaremos si las variantes que tienen en común son correctas o no.

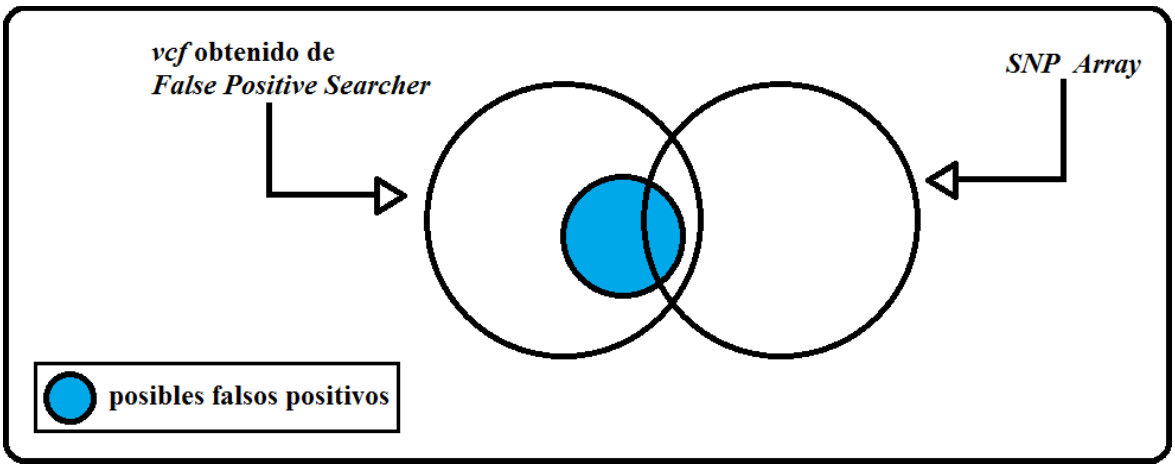

Figura 4.8: *intersección entre el fichero obtenido de False Positive Searcher y el SNP Array.*

A continuación podemos ver los resultados obtenidos tras hacer la intersección entre ficheros de ***False Positive Searcher*** para cinco pacientes y su SNP Array correspondiente:

SNP Array	0/1	1/1
0/0	6	7960
0/1	7363	2502
1/1	4	2774

Tabla 4.9: *resultados de la intersección del fichero obtenido de False Positive Searcher para el paciente 1 y su correspondiente SNP Array.*

SNP Array	0/1	1/1
0/0	17	7089
0/1	7619	2431
1/1	7	2821

Tabla 4.10: *resultados de la intersección del fichero obtenido de False Positive Searcher para el paciente 2 y su correspondiente SNP Array.*

SNP Array	0/1	1/1
0/0	8	6707
0/1	6462	1976
1/1	2	2294

Tabla 4.11: *resultados de la intersección del fichero obtenido de False Positive Searcher para el paciente 3 y su correspondiente SNP Array.*

SNP Array	0/1	1/1
0/0	11	9506
0/1	7768	2831
1/1	8	3907

Tabla 4.12: *resultados de la intersección del fichero obtenido de False Positive Searcher para el paciente 4 y su correspondiente SNP Array.*

SNP Array	0/1	1/1
0/0	17	8711
0/1	7766	2858
1/1	5	3164

Tabla 4.13: *resultados de la intersección del fichero obtenido de False Positive Searcher para el paciente 5 y su correspondiente SNP Array.*

En las tablas podemos observar el número de falsos positivos encontrados señalados en color rojo y el número de verdaderos positivos marcados en color verde.

En color amarillo nos encontramos el número de casos que comparten la situación descrita en el apartado 4.4.1 (hipótesis B) para las muestras (ver código 4.14).

En los resultados vemos que se encuentran muy pocos casos para los que la variante resulta ser homocigota silvestre (0/0) en lugar de ser heterocigota (0/1). Precisamente, este tipo de falsos positivos son los que tratábamos de localizar con ***False Positive Searcher***, ya que partíamos de la hipótesis de que era en las variantes pertenecientes a este grupo donde se encontraban las más susceptibles de ser falsos positivos [22, 23 y 24].

De hecho, para cada paciente, la herramienta indicó que existían entre unos 6000 y 8000 posibles falsos positivos de tipo solapamiento (marcados con la etiqueta ***FP***), y tras la comprobación vemos que realmente unos pocos son realmente falsos positivos.

Sin embargo, sí encontramos un gran número de variantes que en principio se presuponen homocigotas alternas (1/1) cuando realmente son homocigotas silvestres (0/0).

Posteriormente, se intentó identificar este último tipo de falsos positivos utilizando los mismos valores que empleamos para hacer el análisis con ***False Positive Searcher*** (GT, PL, GQ y QD), pero vimos que en base a los valores que nos dan estos campos, resulta muy complicado distinguir entre ambos tipos de variantes, pues la distribución de estos valores para las mismas es demasiado homogénea.

5. CONCLUSIONES Y TRABAJOS FUTUROS

Como comentarios finales sobre las utilidades desarrolladas, podemos concluir lo siguiente:

Por un lado, tenemos la herramienta ***Combinator***, que cumple satisfactoriamente con los objetivos para los que fue diseñada:

1. Combinar variantes de varias muestras (pacientes y controles).

2. Incorporar la funcionalidad de identificar variantes que se encuentran en una zona de baja cobertura (MIST).

3. Añadir información adicional proporcionada por anotadores.

Como trabajo futuro, quedaría su integración en la aplicación global utilizada por la UICHUIMI (*COAT - Combination and Annotation Tool*). También es interesante cualquier mejora en cuanto a rendimiento que se pueda llevar a cabo.

Por otro lado, tenemos la herramienta ***False Positive Searcher***, que tal como vimos en el apartado 4.4.2, no identifica apropiadamente los falsos positivos de tipo solapamiento utilizando únicamente los campos de probabilidades y calidades que tenemos disponibles en un fichero vcf. Al mismo tiempo, nos hemos encontrado otro tipo de falsos positivos, del que aparece un gran número, y para el que resulta muy difícil hacer una detección empleando dichas probabilidades y calidades.

Por lo tanto, como trabajo futuro, habría que tratar de encontrar un método fiable de identificación de falsos positivos, ya que el propuesto en este TFT no ha dado los resultados esperados. En [22] podemos ver otro método de detección de falsos positivos.

Hemos de tener en cuenta que el problema podríamos encontrarlo en que en el fichero vcf (e incluso en otro tipo de archivos utilizados en etapas anteriores a la obtención de un vcf) no se tiene información suficiente para hacer una separación confiable entre variantes reales y falsos positivos.

Una posible causa para el problema de los falsos positivos podemos tenerla en las herramientas y procesos de alineamiento y/o secuenciación en los laboratorios, donde es posible que se produzca un error de identificación de variantes o que la máquina secuenciadora no lea bien las bases, y que este error se vaya arrastrando hacia posteriores etapas. Nos podemos llegar a encontrar ficheros vcf en los que para muestras para las que se sabe que son falsos positivos, tenemos un valor de calidad de lectura alto. Como consecuencia, otro trabajo futuro podría ser analizar estas herramientas y procesos en busca de la causa del error.

Además, es deseable que todo este tipo de utilidades (como las desarrolladas en este TFT) para el análisis y manipulación de información genética, puedan llegar a ser parte de un paquete que ofrezca al usuario las funcionalidades que necesita, y que a su vez, evite la dependencia de programas externos.

6. NORMATIVA Y LEGISLACIÓN

6.1 LEY DE PROTECCIÓN DE DATOS

La razón por la que entra en juego la Ley de Protección de Datos es que las herramientas desarrolladas utilizan como entradas una serie de ficheros que contienen información genética. Estos archivos proceden de muestras extraídas a determinados pacientes, por lo que es necesario proteger cualquier información correspondiente a dichos individuos.

Esta información de carácter personal solamente podrá ser empleada para finalidades relacionadas con el objeto de las investigaciones. La UICHUIMI tiene permiso de los pacientes por escrito y del Comité de Ética del Complejo Hospitalario Universitario Insular Materno Infantil para analizar los datos pertenecientes a los pacientes.

7. REFERENCIAS

[1] National Human Genome Research Institute (NHGRI). An Overview of the Human Genome Project. URL: https://www.genome.gov/12011238/an-overview-of-the-human-genome-project/

[2] Carnevale A. “El nuevo abordaje de las enfermedades mendelianas”. Revista Digital Universitaria (Universidad Nacional Autónoma de México). 1 de junio de 2014, Vol. 15, No.6.

[3] The European Bioinformatics Institute (EMBL-EBI). What is Next-Generation DNA Sequencing. URL: http://www.ebi.ac.uk/training/online/course/ebi-next-generation-sequencing-practical-course/what-you-will-learn/what-next-generation-dna-

[4] Miller SA, Dykes DD y Polesky HF. “A simple salting out procedure for extracting DNA from human nucleated cells”. Nucleic Acids Research 16.3 (feb. 1988): 1215.

[5] Beijing Genomics Institute (BGI). URL: http://www.genomics.cn/en/index

[6] Centro Nacional de Análisis Genómico (CNAG). URL: http://www.cnag.cat/

[7] Cock PJ, Fields CJ, Goto N, Heuer ML, Rice PM. “The Sanger FASTQ File Format for Sequences with Quality Scores, and the Solexa/Illumina FASTQ Variants”. Nucleic Acids Research 38.6 (2010): 1767–1771.

[8] The Genome Reference Consortium. URL: https://www.ncbi.nlm.nih.gov/grc

[9] Li H, Durbin R. “Fast and Accurate Short Read Alignment with Burrows–Wheeler Transform”. Bioinformatics 25.14 (2009): 1754–1760.

[10] Bowtie 2. URL: http://bowtie-bio.sourceforge.net/bowtie2/index.shtml

[11] Short Oligonucleotide Analysis Package (SOAP). URL: http://soap.genomics.org.cn/

[12] Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, Durbin R; 1000 Genome Project Data Processing Subgroup. “The Sequence Alignment/Map format and SAMtools”. Bioinformatics 2009;25:2078-2079. URL: https://samtools.github.io/hts-specs/

[13] Van der Auwera GA, Carneiro MO, Hartl C, Poplin R, Del Angel G, Levy-Moonshine A, Jordan T, Shakir K, Roazen D, Thibault J, Banks E, Garimella KV, Altshuler D, Gabriel S, DePristo MA. “From FastQ data to high confidence variant calls: the Genome Analysis Toolkit best practices pipeline”. Curr Protoc Bioinformatics. 2013;11:11.10.1- 11.10.33. URL: https://software.broadinstitute.org/gatk/

[14] The 1000 Genomes Project Consortium. “A map of human genome variation from population-scale sequencing”. Nature 2010; 467:1061–1073.

[15] Sunyaev S, Ramensky V, Koch I, Lathe W 3rd, Kondrashov AS, Bork P. “Prediction of deleterious human alleles”. Hum Mol Genet. 2001;10:591-597.

[16] McLaren W, Pritchard B, Rios D, Chen Y, Flicek P, Cunningham F. “Deriving the consequences of genomic variants with the Ensembl API and SNP Effect Predictor”. Bioinformatics. 2010;26:2069-2070. URL: http://www.ensembl.org/Tools/VEP

[17] Ng PC, Henikoff S. “SIFT: predicting amino acid changes that affect protein function”. Nucleic Acids Res. 2003;31:3812-3814. URL: http://sift.bii.a-star.edu.sg/

[18] Wang K, Li M, Hakonarson H. “ANNOVAR: functional annotation of genetic variants from high-throughput sequencing data”. Nucleic Acids Research. 2010;38(16):e164. doi:10.1093/nar/gkq603.
URL: http://wannovar.wglab.org/

[19] Lorente-Arencibia P, Guacaran D, Tugores A. “Evaluating the genetic diagnostic power of exome sequencing: Identifying missing data”. doi: https://doi.org/10.1101/068825

[20] Fuentes Fajardo KV, Adams D; NISC Comparative Sequencing Program., Mason CE, Sincan M, Tifft C, Toro C, Boerkoel CF, Gahl W, Markello T. “Detecting False Positive Signals in Exome Sequencing”. Human Mutation 33.4 (2012): 609–613.

[21] Broad Institute. Integrative Genomics Viewer (IGV). URL: http://software.broadinstitute.org/software/igv/

[22] Durtschi J, Margraf RL, Coonrod EM, Mallempati KC, Voelkerding KV. “VarBin, a novel method for classifying true and false positive variants in NGS data”. BMC Bioinformatics 14 (2013) Suppl 13:S2.

[23] Wall JD, Tang LF, Zerbe B, Kvale MN, Kwok PY, Schaefer C, Risch N. “Estimating genotype error rates from high-coverage next-generation sequence data”. Genome Research (2014) 24: 1734-1739.

[24] Kamphans T, Sabri P, Zhu N, Heinrich V, Mundlos S, Robinson PN, Parkhomchuk D, Krawitz PM. “Filtering for Compound Heterozygous Sequence Variants in Non-Consanguineous Pedigrees”. PLOS ONE 10.1371 (2013).

[25] Ewing B, Hillier L, Wendl MC, Green P. “Base-calling of automated sequencer traces using Phred. I. Accuracy assessment”. Genome Research (1998) 8: 175-185.

[26] Ewing B, Green P. “Base-calling of automated sequencer traces using Phred. II. Error Probabilities”. Genome Research (1998) 8: 186-194.

[27] National Human Genome Research Institute (NHGRI). Glosario de términos genéticos. URL: https://www.genome.gov/glossarys/

8. ANEXO: FICHEROS

8.1 *VARIANT CALL FORMAT* (VCF)

Formato genérico para almacenar información relativa al ADN como variantes estructurales y distintas anotaciones [12].

En un fichero de este tipo nos encontramos con dos clases de líneas:

1. *Líneas de cabecera:* comienzan por el símbolo almohadilla '**#**' y describen las etiquetas y anotaciones que aparecen en las líneas de datos.

2. *Líneas de datos:* contienen información y anotaciones para diferentes posiciones en el genoma. Estas líneas se encuentran divididas en una serie de campos.

Campo	Descripción
CHROM	Cromosoma.
POS	Posición dentro del cromosoma.
ID	Identificador único para la variante.
REF	Alelo de referencia.
ALT	Lista de alelos alternos (variantes) separados por coma '**,**'.
QUAL	Valor de calidad.
FILTER	Filtros aplicados.
INFO	Lista de anotaciones separadas por punto y coma '**;**'.
FORMAT	(Campo opcional) indica el formato del campo SAMPLE mediante etiquetas separadas por dos puntos '**:**'.
SAMPLE(s)	(Campo/s opcional/es) una o varias columnas que representan la información especificada en FORMAT para una o varias muestras.

Tabla 8.1: *descripción de los campos principales de un fichero en formato vcf.*

A continuación podemos ver un ejemplo de fichero en formato vcf:

```
##fileformat=VCFv4.1
##ApplyRecalibration="analysis_type=ApplyRecalibration input_file=[] read_buffer_size=null phone_home...
##FILTER=<ID=LowQual,Description="Low quality">
##FILTER=<ID=VQSRTrancheINDEL99.00to99.90,Description="Truth sensitivity tranche level for INDEL...
##...
##FILTER=<ID=VQSRTrancheSNP99.90to100.00,Description="Truth sensitivity tranche level for SNP model...
##FORMAT=<ID=AD,Number=.,Type=Integer,Description="Allelic depths for the ref and alt alleles in the...
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Approximate read depth (reads with MQ=255 or with...
##...
##FORMAT=<ID=PL,Number=G,Type=Integer,Description="Normalized, Phred-scaled likelihoods for genotypes...
##INFO=<ID=AC,Number=A,Type=Integer,Description="Allele count in genotypes, for each ALT allele, in...
##INFO=<ID=AF,Number=A,Type=Float,Description="Allele Frequency, for each ALT allele, in the same...
##...
##INFO=<ID=culprit,Number=1,Type=String,Description="The annotation which was the worst performing in...
##UnifiedGenotyper="analysis_type=UnifiedGenotyper input_file=[alignments/niv_084_alignments.bam]...
##contig=<ID=1,length=249250621,assembly=b37>
##contig=<ID=2,length=243199373,assembly=b37>
##...
##contig=<ID=22,length=51304566,assembly=b37>
##contig=<ID=X,length=155270560,assembly=b37>
##contig=<ID=Y,length=59373566,assembly=b37>
##contig=<ID=MT,length=16569,assembly=b37>
##contig=<ID=GL000207.1,length=4262,assembly=b37>
##...
##contig=<ID=GL000192.1,length=547496,assembly=b37>
##reference=file:///run/media/uai/DATOS/Pascual/genoma/human_g1k_v37.fasta
#CHROM  POS     ID            REF   ALT   QUAL     FILTER                          INFO ...
1       14930   rs75454623    A     G     47.77    VQSRTrancheSNP99.90to100.00   AC=1;AF=0.500;AN=2;...
1       14933   .             G     A     31.77    VQSRTrancheSNP99.90to100.00   AC=1;AF=0.500;AN=2;...
1       14976   rs71252251    G     A     37.77    VQSRTrancheSNP99.90to100.00   AC=1;AF=0.500;AN=2;...
1       17538   .             C     A     205.77   VQSRTrancheSNP99.90to100.00   AC=1;AF=0.500;AN=2;...
1       57856   .             T     A     36.74    VQSRTrancheSNP99.90to100.00   AC=2;AF=1.00;AN=2;...
1       63268   rs75478250    T     C     43.28    VQSRTrancheSNP99.90to100.00   AC=2;AF=1.00;AN=2;DB;...
1       63671   rs116440577   G     A     505.77   VQSRTrancheSNP99.00to99.90     AC=1;AF=0.500;AN=2;...
1       63697   .             T     C     348.77   VQSRTrancheSNP99.90to100.00   AC=1;AF=0.500;AN=2;...
1       120983  rs182468771   C     T     106.03   VQSRTrancheSNP99.90to100.00   AC=2;AF=1.00;AN=2;DB;...
1       121009  rs1851943     C     T     109.03   VQSRTrancheSNP99.00to99.90     AC=2;AF=1.00;AN=2;DB;...
```

Figura 8.1: *ejemplo de fichero en formato vcf.*

8.2 SALIDA DE COMBINATOR (1). COMBINACIÓN DE VARIANTES

```
##fileformat=VCFv4.1
##ApplyRecalibration="analysis_type=ApplyRecalibration input_file=[] read_buffer...
##FILTER=<ID=LowQual,Description="Low quality">
##FILTER=<ID=VQSRTrancheINDEL99.00to99.90,Description="Truth sensitivity tranche...
##...
##FILTER=<ID=VQSRTrancheSNP99.90to100.00,Description="Truth sensitivity tranche...
##INFO=<ID=AF,Number=A,Type=Float,Description="Allele Frequency, for each ALT...
##INFO=<ID=DP,Number=1,Type=Integer,Description="Approximate read depth; some reads...
##UnifiedGenotyper="analysis_type=UnifiedGenotyper input_file=[alignments/niv_084...
##contig=<ID=1,length=249250621,assembly=b37>
##contig=<ID=2,length=243199373,assembly=b37>
##...
##contig=<ID=21,length=48129895,assembly=b37>
##contig=<ID=22,length=51304566,assembly=b37>
##contig=<ID=X,length=155270560,assembly=b37>
##contig=<ID=Y,length=59373566,assembly=b37>
##contig=<ID=MT,length=16569,assembly=b37>
##reference=file:///run/media/uai/DATOS/Pascual/genoma/human_g1k_v37.fasta
##Reference_Include=File:/home/uai02/Investigacion_Jacob/ficheros_vcf_mist/niv_084.vcf
##Include=File:/home/uai02/Investigacion_Jacob/ficheros_vcf_mist/niv_85.vcf
##Exclude=File:/home/uai02/Investigacion_Jacob/ficheros_vcf_mist/sqz_001.vcf
##Exclude=File:/home/uai02/Investigacion_Jacob/ficheros_vcf_mist/wdh_001.vcf
#CHROM  POS      ID           REF   ALT   QUAL     FILTER   INFO
1       808631   rs11240779   G     A     211.77   PASS     AF=0.500;DP=18
1       908823   rs28687780   G     A     193.77   PASS     AF=0.500;DP=19
1       909238   rs3829740    G     C     191.77   PASS     AF=0.500;DP=17
1       909309   rs3829738    T     C     290.77   PASS     AF=0.500;DP=18
1       970722   .            C     T     884.77   PASS     AF=0.500;DP=64
1       974570   rs2465134    T     G     45.74    PASS     AF=1.00;DP=3
1       977203   rs3121552    G     C     78.28    PASS     AF=1.00;DP=3
1       985797   rs142286944  A     G     111.77   PASS     AF=0.500;DP=17
1       1115510  rs114359609  C     T     139.77   PASS     AF=0.500;DP=9
1       1119657  rs4560982    G     C     40.77    PASS     AF=0.500;DP=3
```

Figura 8.2: *ejemplo de fichero de salida de Combinator (1). Combinar variantes de varias muestras*

8.3 *MISSING SEQUENCING TOOLS (*MIST)

chrom	exon_start	exon_end	poor_start	poor_end	gene_id	gene_name	...
1	11869	12227	1	12093	ENSG00000223972	DDX11L1	...
1	11872	12227	1	12093	ENSG00000223972	DDX11L1	...
1	11874	12227	1	12093	ENSG00000223972	DDX11L1	...
1	12010	12057	1	12093	ENSG00000223972	DDX11L1	...
1	13225	14412	13598	14182	ENSG00000223972	DDX11L1	...
1	13225	14412	14199	14209	ENSG00000223972	DDX11L1	...
1	13225	14412	14218	14221	ENSG00000223972	DDX11L1	...
1	13403	13655	13598	14182	ENSG00000223972	DDX11L1	...
1	13453	13670	13598	14182	ENSG00000223972	DDX11L1	...
1	13661	14409	13598	14182	ENSG00000223972	DDX11L1	...

Figura 8.3: *ejemplo de fichero mist.*

8.4 SALIDA DE COMBINATOR (2). IDENTIFICACIÓN DE VARIANTES EN ZONAS MIST

```
##fileformat=VCFv4.1
##ApplyRecalibration="analysis_type=ApplyRecalibration input_file=[] read_buffer_size=null phone...
##FILTER=<ID=LowQual,Description="Low quality">
##FILTER=<ID=VQSRTrancheINDEL99.00to99.90,Description="Truth sensitivity tranche level for INDEL...
##...
##FILTER=<ID=VQSRTrancheSNP99.90to100.00,Description="Truth sensitivity tranche level for SNP...
##INFO=<ID=MistZone,Number=0,Type=Flag,Description="If present, indicates that the position is...
##INFO=<ID=AF,Number=A,Type=Float,Description="Allele Frequency, for each ALT allele, in the sam...
##INFO=<ID=DP,Number=1,Type=Integer,Description="Approximate read depth; some reads may have...
##UnifiedGenotyper="analysis_type=UnifiedGenotyper input_file=[alignments/niv_084_alignments.bam]...
##contig=<ID=1,length=249250621,assembly=b37>
##contig=<ID=2,length=243199373,assembly=b37>
##...
##contig=<ID=22,length=51304566,assembly=b37>
##contig=<ID=X,length=155270560,assembly=b37>
##contig=<ID=Y,length=59373566,assembly=b37>
##contig=<ID=MT,length=16569,assembly=b37>
##reference=file:///run/media/uai/DATOS/Pascual/genoma/human_g1k_v37.fasta
##Reference_Include=File:/home/uai02/Investigacion_Jacob/ficheros_vcf_mist/niv_084.vcf
##Include=File:/home/uai02/Investigacion_Jacob/ficheros_vcf_mist/niv_85.vcf
##Exclude=File:/home/uai02/Investigacion_Jacob/ficheros_vcf_mist/sqz_001.vcf
##Exclude=File:/home/uai02/Investigacion_Jacob/ficheros_vcf_mist/wdh_001.vcf
#CHROM  POS     ID          REF  ALT  QUAL     FILTER                      INFO
1       808631  rs11240779  G    A    211.77   PASS                        AF=0.500;DP=18
...
1       1149316 rs35737009  C    G    86.77    PASS                        AF=0.500;DP=6
1       1179385 .           A    G    35.77    VQSRTrancheSNP99.00to99.90  MistZone;AF=0.500;DP=4
1       1179902 .           G    A    216.77   PASS                        AF=0.500;DP=19
1       1220954 rs12751100  G    A    34.77    PASS                        MistZone;AF=0.500;DP=3
...
1       1247302 rs146699248 G    A    73.77    PASS                        MistZone;AF=0.500;DP=8
1       1262957 rs140494438 C    T    177.82   PASS                        AF=0.500;DP=8
1       1268159 rs3813210   C    T    64.77    PASS                        MistZone;AF=0.500;DP=6
1       1275716 rs112871502 T    A    222.77   PASS                        AF=0.500;DP=17
1       1314172 rs2477777   C    T    77.28    PASS                        MistZone;AF=1.00;DP=3
```

Figura 8.4: *ejemplo de fichero de salida de Combinator (2). Identificar variantes que se encuentran en una zona mist.*

8.5 *VARIANT EFFECT PREDICTOR* (VEP)

#Uploaded_variation	Location	Allele	Gene	Feature	Feature_type	...
rs11240779	1:808631	A	ENSG00000230368	ENST00000427857	Transcript	...
rs11240779	1:808631	A	ENSG00000234711	ENST00000415481	Transcript	...
rs11240779	1:808631	A	ENSG00000230368	ENST00000432963	Transcript	...
rs11240779	1:808631	A	ENSG00000230368	ENST00000446136	Transcript	...
rs11240779	1:808631	A	-	ENSR00001516729	RegulatoryFeature	...
rs28687780	1:908823	A	ENSG00000187642	ENST00000341290	Transcript	...
rs28687780	1:908823	A	ENSG00000187642	ENST00000433179	Transcript	...
rs28687780	1:908823	A	ENSG00000187583	ENST00000491024	Transcript	...
rs28687780	1:908823	A	ENSG00000187583	ENST00000379409	Transcript	...
rs28687780	1:908823	A	ENSG00000187583	ENST00000480267	Transcript	...

Figura 8.5: *ejemplo de fichero obtenido del anotador VEP.*

8.6 *SORT INTOLERANT FROM TOLERANT* (SIFT)

Coordinates	Codons	Ensembl Transcript ID	RefSeq Transcript ID	Known Transcript ID	...
1,909238,1,G/C	CGT-CcT	ENST00000379407	NM_001160184	uc001acf.2	...
1,909309,1,T/C	TCC-cCC	ENST00000379407	NM_001160184	uc001acf.2	...
1,1115510,1,C/T	CCG-CtG	ENST00000379288	NM_153254	uc001acz.1	...
1,1120431,1,G/A	AGT-AaT	ENST00000379288	NM_153254	uc001acz.1	...
1,1179385,1,A/G	GGT-GGc	ENST00000330388	NM_001014980	uc001adl.1	...
1,1220954,1,G/A	GCC-aCC	ENST00000379110			...
1,1221001,1,A/G	GCA-GCg	ENST00000379110			...
1,1247302,1,G/A	GAC-GAt	ENST00000419704		uc001aeh.1	...
1,1262957,1,C/T	GTC-GTt	ENST00000343938	NM_001029885	uc001aeo.2	...
1,1268159,1,C/T	CCC-CCt	ENST00000339381	NM_152228		...

Figura 8.6: *ejemplo de fichero obtenido del anotador SIFT.*

8.7 *ANNOTATE VARIATION* (ANNOVAR)

Chr,	Start,	End,	Ref,	Alt,	Func.refgene,	Gene.refgene,	...
1,	808631,	808631,	G,	A,	ncRNA_intronic,	FAM41C,	...
1,	908823,	908823,	G,	A,	intronic,	PLEKHN1,	...
1,	909238,	909238,	G,	C,	exonic,	PLEKHN1,	...
1,	909309,	909309,	T,	C,	exonic,	PLEKHN1,	...
1,	970722,	970722,	C,	T,	intronic,	AGRN,	...
1,	974570,	974570,	T,	G,	intronic,	AGRN,	...
1,	977203,	977203,	G,	C,	intronic,	AGRN,	...
1,	985797,	985797,	A,	G,	intronic,	AGRN,	...
1,	1115510,	1115510,	C,	T,	exonic,	TTLL10,	...
1,	1119657,	1119657,	G,	C,	intronic,	TTLL10,	...

Figura 8.7: *ejemplo de fichero obtenido del anotador ANNOVAR.*

8.8 SALIDA DE COMBINATOR (3). ANOTACIÓN DE VARIANTES

```
##fileformat=VCFv4.1
##ApplyRecalibration="analysis_type=ApplyRecalibration input_file=[] read_buffer_size=null phone...
##FILTER=<ID=LowQual,Description="Low quality">
##FILTER=<ID=VQSRTrancheINDEL99.00to99.90,Description="Truth sensitivity tranche level for INDEL...
##...
##FILTER=<ID=VQSRTrancheSNP99.90to100.00,Description="Truth sensitivity tranche level for SNP...
##INFO=<ID=MistZone,Number=0,Type=Flag,Description="If present, indicates that the position is...
##INFO=<ID=AF,Number=A,Type=Float,Description="Allele Frequency, for each ALT allele, in the same...
##...
##INFO=<ID=SOMA,Number=1,Type=String,Description="Somatic status of existing variation(s)">
##UnifiedGenotyper="analysis_type=UnifiedGenotyper input_file=[alignments/niv_084_alignments.bam]...
##contig=<ID=1,length=249250621,assembly=b37>
##contig=<ID=2,length=243199373,assembly=b37>
##...
##contig=<ID=22,length=51304566,assembly=b37>
##contig=<ID=X,length=155270560,assembly=b37>
##contig=<ID=Y,length=59373566,assembly=b37>
##contig=<ID=MT,length=16569,assembly=b37>
##reference=file:///run/media/uai/DATOS/Pascual/genoma/human_g1k_v37.fasta
##Reference_Include=File:/home/uai02/Investigacion_Jacob/ficheros_vcf_mist/niv_084.vcf
##Include=File:/home/uai02/Investigacion_Jacob/ficheros_vcf_mist/niv_85.vcf
##Exclude=File:/home/uai02/Investigacion_Jacob/ficheros_vcf_mist/sqz_001.vcf
##Exclude=File:/home/uai02/Investigacion_Jacob/ficheros_vcf_mist/wdh_001.vcf
#CHROM  POS     ID          REF  ALT  QUAL    FILTER  INFO ...
1       808631  rs11240779  G    A    211.77  PASS    AF=0.500;BIO=lincRNA;CONS=intron_variant,...
1       908823  rs28687780  G    A    193.77  PASS    AF=0.500;AFR_F=0.07;AMR_F=0.18;ASN_F=0.08;...
1       909238  rs3829740   G    C    191.77  PASS    AA_F=0.780838;AF=0.500;AFR_F=0.81;AMR_F=...
1       909309  rs3829738   T    C    290.77  PASS    AA_F=0.242610;AF=0.500;AFR_F=0.21;AMR_F=...
1       970722  .           C    T    884.77  PASS    AA_F=0.000227;AF=0.500;BIO=protein_coding;...
1       974570  rs2465134   T    G    45.74   PASS    AF=1.00;AFR_F=0.69;AMR_F=0.93;BIO=protein...
1       977203  rs3121552   G    C    78.28   PASS    AF=1.00;AFR_F=0.69;AMR_F=0.93;BIO=protein...
1       985797  rs142286944 A    G    111.77  PASS    AA_F=0.007947;AF=0.500;AFR_F=0.0020;AMR_F=...
1       1115510 rs114359609 C    T    139.77  PASS    AA_F=0.012483;AF=0.500;AFR_F=0.01;AMR_F=...
1       1119657 rs4560982   G    C    40.77   PASS    AF=0.500;AFR_F=0.96;AMR_F=0.57;ASN_F=0.91;...
```

Figura 8.8: *ejemplo de fichero de salida de Combinator (3). Anotar las variantes resultantes. En este ejemplo se han añadido las anotaciones del anotador VEP.*

8.9 SALIDA DE FALSE POSITIVE SEARCHER

```
##fileformat=VCFv4.1
##reference=file:///home/unidad03/DNA_Sequencing/HomoSapiensGRCh37/human_g1k_v37.fasta
##FILTER=<ID=LowQual,Description="Low quality">
##FORMAT=<ID=GT,Description=Genotype,Number=1,Type=String>
##...
##FORMAT=<ID=PL,Description="Normalized, Phred-scaled likelihoods for genotypes as defined in the VCF spe...
##GATKCommandLine=<ID=HaplotypeCaller,CommandLineOptions="analysis_type=HaplotypeCaller input_file=[/home...
##INFO=<ID=AC,Description="Allele count in genotypes, for each ALT allele, in the same order as listed",N...
##INFO=<ID=AF,Description="Allele Frequency, for each ALT allele, in the same order as listed",Number=A,T...
##...
##INFO=<ID=SOR,Description="Symmetric Odds Ratio of 2x2 contingency table to detect strand bias",Number=1...
##INFO=<ID=FP,Number=0,Type=Flag,Description="If present, indicates that the variant is likely a false...
##contig=<ID=1,assembly=b37,length=249250621>
##contig=<ID=2,assembly=b37,length=243199373>
##...
##contig=<ID=GL000194.1,assembly=b37,length=191469>
##contig=<ID=GL000225.1,assembly=b37,length=211173>
##contig=<ID=GL000192.1,assembly=b37,length=547496>
#CHROM  POS     ID          REF   ALT   QUAL         FILTER   INFO ...
1       13273   .           G     C     124.77       .           AC=1;AF=0.500;FP;AN=2;BaseQRankSum=0.972;...
1       69511   rs75062661  A     G     1592.7700    .           AC=2;AF=1.00;AN=2;DB;DP=60;FS=0.000;MLEAC...
1       133160  .           G     A     118.7700     .           AC=1;AF=0.500;AN=2;BaseQRankSum=-1.221;...
1       139213  .           A     G     67.77        .           AC=1;AF=0.500;FP;AN=2;BaseQRankSum=0.296;...
1       139233  .           C     A     69.77        .           AC=1;AF=0.500;FP;AN=2;BaseQRankSum=1.683;...
1       651149  .           C     T     40.7400      .           AC=2;AF=1.00;AN=2;DP=2;FS=0.000;MLEAC=2;...
1       715348  rs3131984   T     G     85.2800      .           AC=2;AF=1.00;AN=2;DB;DP=3;FS=0.000;MLEAC=...
1       752566  rs3094315   G     A     190.8400     .           AC=2;AF=1.00;AN=2;DB;DP=6;FS=0.000;MLEAC=...
1       752721  rs3131972   A     G     1228.7700    .           AC=2;AF=1.00;AN=2;DB;DP=38;FS=0.000;MLEAC...
1       752894  rs3131971   T     C     440.7700     .           AC=2;AF=1.00;AN=2;DB;DP=15;FS=0.000;MLEAC...
```

Figura 8.9: *ejemplo de fichero de salida de False Positive Searcher.*

8.10 SNP ARRAY

```
##fileformat=VCF4.2
##fileDate=20151117
##source=PLINKv1.90
##contig=<ID=0,length=2147483645>
##contig=<ID=1,length=249212216>
##...
##contig=<ID=20,length=62904703>
##contig=<ID=21,length=48084248>
##contig=<ID=22,length=51162060>
##...
##INFO=<ID=PR,Number=0,Type=Flag,Description="Provisional reference allele, may not be based on...
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
#CHROM  POS     ID          REF  ALT  QUAL  FILTER  INFO  FORMAT  SAMPLE_1  SAMPLE_2  SAMPLE_3 ...
0       0       200610-10   A    .    0     .       PR    GT      0/0       0/0       0/0      ...
0       0       200610-108  A    .    0     .       PR    GT      0/0       0/0       0/0      ...
...
1       569418  MitoA8870G  A    .    0     .       PR    GT      0/0       0/0       0/0      ...
1       752566  rs3094315   A    G    0     .       PR    GT      0/1       0/0       0/1      ...
1       752721  rs3131972   G    A    0     .       PR    GT      0/1       0/0       0/1      ...
1       798959  rs11240777  G    A    0     .       PR    GT      0/1       0/1       0/1      ...
1       838555  rs4970383   C    A    0     .       PR    GT      0/0       0/0       0/1      ...
1       846808  rs4475691   G    A    0     .       PR    GT      0/0       0/0       0/0      ...
1       861808  rs13302982  G    A    0     .       PR    GT      0/0       0/0       0/0      ...
1       887521  variant.16  A    .    0     .       PR    GT      0/0       0/0       0/0      ...
```

Figura 8.10: *ejemplo de fichero SNP Array.*

9. GLOSARIO

[1] **Alelo** [27]**:** cada una de las dos o más versiones de un gen. Un individuo hereda dos alelos para cada gen, uno del padre y el otro de la madre. Los alelos se encuentran en la misma posición dentro de los cromosomas homólogos.

[2] **Bioinformática** [27]**:** subdisciplina de la biología y las ciencias computacionales que se encarga de adquirir, almacenar, analizar y diseminar la información biológica, en gran parte correspondiente a las secuencias de ADN y aminoácidos. La Bioinformática utiliza programas informáticos que tienen muchas aplicaciones, como determinar las funciones de genes y proteínas, establecer relaciones evolutivas y predecir la conformación tridimensional de las proteínas.

[3] **Codón** [27]**:** secuencia de tres nucleótidos de ADN o ARN que corresponde a un aminoácido específico. El código genético describe la relación entre la secuencia de bases del ADN (A, C, G y T) en un gen y la secuencia correspondiente de la proteína que codifica. La célula lee la secuencia del gen en grupos de tres bases. Existen 64 codones diferentes: 61 son específicos de aminoácidos, mientras que los tres restantes se utilizan como señales de parada.

[4] **Enfermedad de Wilson:** trastorno hereditario que causa que haya demasiado cobre en los tejidos del cuerpo. Este exceso de cobre causa daño al hígado y al sistema nervioso.

[5] **Enfermedades mendelianas** [2]**:** son aquellas causadas por mutaciones en los genes. Las mutaciones son variantes en los genes, capaces de alterar la función de la proteína codificada por el gen, de forma que no se produce, lo hace en cantidad disminuida o aumentada, es inestable, no funciona

adecuadamente o se expresa de manera inapropiada. Estas variantes genéticas producen defectos o enfermedades.

[6] **Exón** [27]**:** porción de gen que codifica aminoácidos. Son las partes de la secuencia de genes que contienen la información para producir las proteínas, mientras que los intrones son las partes de la secuencia del gen que no codifican y se encuentran en medio o interfieren con los exones.

[7] **Gen** [27]**:** unidad física básica de la herencia. Los genes se transmiten de los padres a la descendencia y contienen la información necesaria para precisar sus rasgos. Están dispuestos, uno tras otro, en estructuras llamadas cromosomas. Un cromosoma contiene una única molécula larga de ADN, de la cual una parte corresponde a un gen individual. Los seres humanos tienen aproximadamente 20.000 genes organizados en sus cromosomas.

[8] **Genoma** [27]**:** conjunto de instrucciones genéticas que se encuentra en una célula. En los seres humanos, el genoma consiste en 23 pares de cromosomas, que se encuentran en el núcleo, así como un pequeño cromosoma que se encuentra en las mitocondrias de las células.

[9] **Heterocigoto** [27]**:** un individuo que hereda los dos alelos diferentes, uno del padre y otro de la madre, es heterocigoto para ese gen.

[10] **Homocigoto** [27]**:** un individuo que hereda los dos alelos idénticos, uno del padre y otro de la madre, es homocigoto para ese gen.

[11] **Mutación endémica:** enfermedad que se desarrolla habitualmente en una región determinada.

[12] **Nucleótido** [27]**:** pieza básica de los ácidos nucleicos. El ADN y el ARN son polímeros formados por largas cadenas de nucleótidos. Un nucleótido está formado por una molécula de azúcar (desoxirribosa en el ADN o ribosa en el ARN) unido a un grupo fosfato y una base nitrogenada. Las bases

utilizadas en el ADN son la adenina (A), citosina (C), guanina (G) y timina (T). En el ARN, la base uracilo (U) ocupa el lugar de la timina.

[13] **Polímero:** macromolécula formada por la unión de moléculas más pequeñas llamadas monómeros.

[14] **Proteína** [27]**:** una clase importante de moléculas que se encuentran en todas las células vivas. Se compone de una o más cadenas largas de aminoácidos, cuya secuencia corresponde a la secuencia de ADN del gen que la codifica. Las proteínas desempeñan gran variedad de funciones en la célula, incluidas estructurales, mecánicas y bioquímicas.

[15] **Proyecto del Genoma Humano** [1 y 27]**:** proyecto internacional que mapeó y secuenció todos los genes humanos. Terminado en abril de 2003, los datos del proyecto están a libre disposición de los investigadores y otros interesados en genética y salud humana.

[16] **Recesivo** [27]**:** este término se refiere a la relación entre dos versiones de un gen. Los individuos reciben una versión de un gen, llamada alelo, de cada padre. Si los alelos son diferentes, el alelo dominante se expresa, mientras que el efecto del otro alelo, denominado recesivo, queda enmascarado. En el caso de un trastorno genético recesivo, un individuo debe haber heredado las dos copias del alelo mutado para que la enfermedad esté presente.

[17] **Ribosoma** [27]**:** partícula celular hecha de ARN y proteína que sirve como el sitio para la síntesis de proteínas en la célula. El ribosoma lee la secuencia del ARN mensajero y, utilizando el código genético, se traduce la secuencia de bases del ARN a una secuencia de aminoácidos.

[18] **Secuenciación del ADN** [27]**:** identificación de la secuencia exacta de las bases (A, C, G y T) en una molécula de ADN. La secuencia de bases de ADN lleva la información que una célula necesita para ensamblar proteínas y moléculas de ARN. Esta información es importante para los científicos

que investigan las funciones de los genes. La tecnología de secuenciación de ADN se hizo más rápida y menos costosa como resultado del *Proyecto del Genoma Humano*[15].

[19] **SNP Array:** método de detección de variantes basado en hibridación de secuencias tipo inmovilizadas con una sonda fluorescente originada de un individuo. Este método nos da una alta fiabilidad de los datos obtenidos.

[20] **Variante:** diferencia entre el genoma de una persona y un genoma de referencia, ya sea el estándar del Genome Reference Consortium [8] o el de otra persona.