

**INSTITUTO TECNOLÓGICO DE BUENOS AIRES – ITBA**  
**ESCUELA DE INGENIERÍA Y TECNOLOGÍA**

**DESARROLLO DE UN FLUJO**  
**BIOINFORMÁTICO PARA EL ANÁLISIS**  
**GENÉTICO DE RESISTENCIA A DROGAS**  
**DEL VIRUS DE HEPATITIS C MEDIANTE LA**  
**DETECCIÓN DE MUTACIONES DE BAJA**  
**FRECUENCIA**

**AUTOR: Vaca Díez, Gustavo (Leg. N° 53580)**

**TUTOR: Oliver, Javier**

**TRABAJO FINAL PRESENTADO PARA LA OBTENCIÓN DEL TÍTULO DE BIOINGENIERO**

**BUENOS AIRES**  
**SEGUNDO CUATRIMESTRE, 2018**



## Resumen

Nuevas tecnologías de análisis genético han revolucionado el panorama del diagnóstico en la medicina. La secuenciación masiva en paralelo o *Next Generation Sequencing* (NGS), ha logrado llevar la secuenciación del genoma a costos considerablemente menores, y toma cada vez más terreno en el ámbito clínico. El hallazgo de relaciones causales entre variantes genéticas y enfermedades, o características fisiológicas derivadas de las mismas, está abriendo paso a nuevas opciones de tratamiento dirigido en subpoblaciones genéticas particulares, logrando impactos contundentes y minimizando efectos secundarios y riesgo de recaída. En el presente trabajo se analiza cómo se puede contribuir al diseño de estrategias de tratamiento dirigido al secuenciar el genoma de los organismos que provocan las enfermedades, aplicado particularmente al virus de Hepatitis C. Debido a la replicación no regulada del virus, existen altas probabilidades de que se generen variantes asociadas a resistencia a los compuestos usados en los tratamientos, como los antivirales de acción directa o *Direct Acting Anti-virals* (DAAs). Dado que la detección de mutaciones de baja frecuencia es compleja, los pacientes con mayor riesgo de recaída son aquellos que poseen una mínima proporción de la población viral resistente a su tratamiento y que no fue detectada en el diagnóstico inicial. Estas subpoblaciones resistentes se encuentran a baja frecuencia, y en los datos de secuenciación se hallan proporciones entre 1 y 5%. Con el fin de contribuir en las decisiones clínicas de tratamiento, se propone en éste estudio el uso herramientas bioinformáticas para determinar la proporción de variantes asociadas a resistencia en muestras poblacionales de Hepatitis C.

## Agradecimientos

Agradezco a todos los que me ayudaron y apoyaron en este trabajo. Quiero agradecer especialmente a Javier, mi tutor, por haber sido un excelente guía en el proceso de descubrimientos de esta experiencia y por haberme compartido tanto de su tiempo, que es lo más valioso que un ser humano puede dar.

## Prólogo

La actual tesina es presentada como Proyecto Final de Grado de la carrera de Bioingeniería del Instituto Tecnológico de Buenos Aires. Se realizó en el marco de la materia Proyecto Final a cargo del MEE. Ing. Norberto Lerendegui. La tesina fue evaluada por la Dra. Mariela Bollini y la Dra. Maria Laura Fernandez. Fue presentada su versión final el 11 de Septiembre del 2018.

# Glosario

**ADN** abreviación de ácido desoxirribonucleico, es un ácido nucleico que contiene las instrucciones genéticas usadas en el desarrollo y funcionamiento de todos los organismos vivos y algunos virus, también es responsable de la transmisión hereditaria. 3

**antivirales de acción directa** combinación de drogas que atacan directamente al mecanismo de acción viral para prevenir la replicación de su material genético. 20

**ARN** abreviación de ácido ribonucleico, es un ácido nucleico formado por una cadena de ribonucleótidos. 5

**biblioteca** una colección de fragmentos de ADN de tamaño similar con secuencias adaptadoras conocidas añadidas a los extremos 5' y 3'. 6

**CCD** abreviación de dispositivo de carga acoplada, es un circuito integrado que contiene un número determinado de condensadores enlazados o acoplados. 4

**dataframe** un tipo de estructura de datos en el lenguaje de programación R para representar tablas. 29

**farmacogenómica** ciencia que estudia la respuesta a drogas según el perfil genético, a fin de desarrollar drogas y dosis efectivas y seguras, a partir del análisis de grandes grupos de genes, utilizando generalmente tecnología NGS. 2

**FASTA** tipo de archivo basado en texto que registra secuencias de aminoácidos o ácidos nucleidos. 24

**FASTQ** tipo de archivo basado en texto que registra listas de secuencias de aminoácidos o ácidos nucleidos junto con sus calificaciones de calidad de lectura. 9

- Hepatitis C** una enfermedad infecciosa que afecta principalmente al hígado y es causada por el virus de la hepatitis C (VHC). 17
- Illumina** es una compañía estadounidense fundada en abril de 1998 que desarrolla, fabrica y comercializa sistemas integrados para el análisis de variación genética y función biológica. 3
- indel** contracción de "inserción o deleción" y se refiere a mutaciones genéticas caracterizadas por la inserción o deleción de una cantidad baja de nucleótidos (de 1 a 10 000 pares de bases) en la secuencia de ADN. 14
- Ion Torrent** es un método de secuenciación de ADN basado en la detección de protones liberados durante el proceso de polimerización del ADN. 3
- medicina de precisión** estrategia emergente para el tratamiento y la prevención médica, que tiene en cuenta la variabilidad individual en genes, ambiente y forma de vida. 1
- microarrays** es una tecnología en desarrollo para estudiar la expresión de muchos genes a la vez, consiste en colocar miles de secuencias génicas en lugares determinados sobre un portaobjetos de vidrio llamado chip. 10
- MinION** plataforma de secuenciación con un método para determinar el orden en el que los nucleótidos se organizan en una sola hebra de ADN que atraviesa un nanoporo. 4
- mutación de baja frecuencia** mutación que aparece en una proporción poblacional menor al 5%. 21
- PacBio** es una plataforma con tecnología de secuenciación por síntesis de una molécula de ADN fija desarrollada por Pacific Biosciences. 4
- PCR cuantitativa** también llamada PCR en tiempo real (en inglés real-time PCR) es una variante de la reacción en cadena de la polimerasa (PCR) utilizada para amplificar y simultáneamente cuantificar de forma absoluta el producto de la amplificación de ADN. 10
- profundidad** número promedio de reads que representa un nucleótido determinado en la secuencia reconstruida. 8

- Roche 454** es una tecnología que permite determinar una secuencia de ADN a gran escala, aplicable a genomas completos, mediante luminiscencia. 2
- SAM** es un formato basado en texto para almacenar secuencias biológicas alineadas a una secuencia de referencia desarrollada por Heng Li y Bob Handsaker et al.. 12
- Sanger** desarrollado en 1975, es el primer método de secuenciación por dideoxinucléotidos, se basa en el proceso biológico de la replicación del ADN. 10
- secuencia de referencia** es una base de datos digital de secuencias de ácidos nucleicos, ensamblada por científicos como un ejemplo representativo del conjunto de genes de una especie. 2
- secuenciación masiva en paralelo** también conocida como *next generation sequencing* (NGS) consiste en obtener la secuencia de ADN deseada a través de tecnologías de alto rendimiento. 2
- secuenciación por ligación** es un método de secuenciación de ADN que utiliza la enzima ADN ligasa para identificar el nucleótido presente en una posición determinada en una secuencia de ADN. 3
- secuenciación por síntesis** una de las tecnologías de secuenciación masiva de nueva generación que utilizan la amplificación clonal in vitro por medio de un PCR en puente. 3
- SNP** los polimorfismos de nucleótido único o Single Nucleotide Polymorphism (SNP) son variaciones en la secuencia de ADN que afectan a una sola base de una secuencia del genoma. 14
- SV** tipo de variaciones se refiere a duplicaciones, inversiones, inserciones o variantes en el número de copias de segmentos grandes del genoma. 14
- transcripción inversa** es un proceso de la biología molecular que implica la generación de una cadena de ácido desoxirribonucleico (ADN) de doble cadena a partir de un ácido ribonucleico (ARN) de cadena simple. 5
- transcriptasa inversa** es una enzima de tipo ADN-polimerasa, que tiene como función sintetizar ADN de doble cadena utilizando como molde ARN monocatenario, es decir, catalizar la retrotranscripción o transcripción inversa. 5

**transformada de Burrows-Wheeler** es un algoritmo usado en técnicas de compresión de datos como en bzip2, fue inventado por Michael Burrows y David Wheeler en 1994 mientras trabajaban en el DEC Systems Research Center en Palo Alto, California. 12

**tratamiento dirigido** tipo de tratamiento en el que se usa medicamentos u otras sustancias para identificar y combatir tipos específicos de patologías. 21

**VCF** es un fichero de texto que se usa en Bioinformática para almacenar variaciones de la secuencia de genes y su información. 16

# Índice general

<b>1. Introducción</b>	<b>1</b>
1.1. Medicina de Precisión . . . . .	1
1.1.1. Secuenciación masiva en paralelo . . . . .	2
1.1.2. Tecnologías de secuenciación NGS . . . . .	3
1.2. Flujo Bioinformático . . . . .	11
1.2.1. Alineamiento . . . . .	12
1.2.2. Variant Calling . . . . .	13
1.2.3. Formato VCF y Variant Annotation . . . . .	16
1.3. Virus de Hepatitis C . . . . .	17
1.3.1. Genoma y Mecanismo de Acción Viral . . . . .	17
1.3.2. Tratamientos . . . . .	20
1.3.3. Variantes Asociadas a Resistencia . . . . .	20
1.4. Evaluando Variant Callers en el Virus de Hepatitis C . . . . .	21
<b>2. Objetivos</b>	<b>23</b>
2.1. Objetivo General . . . . .	23
2.2. Objetivos Específicos . . . . .	23
<b>3. Materiales y Metodos</b>	<b>24</b>
3.1. Obtención de Datos . . . . .	24
3.1.1. Obtención de datos artificiales con Artificial Fastq Generator . . . . .	25
3.2. Procesamiento Bioinformático de los Datos . . . . .	26
3.2.1. Alineamiento . . . . .	26
3.2.2. Variant Callers . . . . .	27
3.3. Diseño del Naive Variant Caller . . . . .	28
3.3.1. Detector de Cambios de Aminoácidos . . . . .	29
3.4. Evaluación de Variant Callers con Datos Artificiales . . . . .	30
3.5. Evaluación de Variant Callers con Datos Reales . . . . .	33
3.5.1. Gold Standard Basado en Variantes Asociadas de Resistencia . . . . .	33

---

3.5.2. Gold Standard Basado en Variantes con Cambio de Sentido . . .	34
<b>4. Resultados y Discusión</b>	<b>36</b>
4.1. Resultados de Rendimiento de Detección con Datos Artificiales . . . .	36
4.2. Resultados de Rendimiento de Detección con Datos Reales . . . . .	44
4.2.1. Rendimiento con Gold Standard basado en Variantes Asociadas a Resistencia . . . . .	44
4.2.2. Rendimiento con Gold Standard basado en Variantes con Cam- bio de Sentido . . . . .	47
<b>5. Conclusiones</b>	<b>52</b>
<b>A. Información Adicional</b>	<b>54</b>
A.1. Lista de Datos Seleccionados de Base de Datos de NCBI . . . . .	54
A.2. Tablas de Resultados de Evaluación de Datos Artificiales . . . . .	54
A.3. Tablas de Resultados de Evaluación de Datos Reales . . . . .	56
A.3.1. Gold Standard Basado en Variantes Asociadas a Resistencia . .	56
A.3.2. Gold Standard Basado en Variantes con Cambio de Sentido . .	58
<b>B. Programas</b>	<b>61</b>
B.1. Procesamiento en R del Naive Variant Caller . . . . .	61
B.2. Evaluador de Datos Artificiales . . . . .	65
B.3. Evaluador de Datos Reales . . . . .	71
<b>Bibliografía</b>	<b>84</b>
<b>Índice de figuras</b>	<b>91</b>
<b>Índice de cuadros</b>	<b>93</b>

*«Solíamos pensar que nuestro futuro se encontraría en las estrellas. Ahora sabemos que está en nuestros genes»*

— James Watson

*«para mi padre, que siempre me inspiró a tener grandes sueños y a mi madre, que me preparó para alcanzarlos. A Jessica, que sin su compañía estos sueños no tendrían sentido.»*

# 1. Introducción

«La patología celular no es ningún fin por si mismo, si no se puede apreciar ninguna alteración en la célula. La química trae la clarificación de los procesos vivientes más cerca que la anatomía. Cada cambio anatómico debe haber sido precedido por uno químico.»

— Rudolf Virchow

## 1.1. Medicina de Precisión

El uso del término “medicina de precisión” comenzó a tomar fuerza con una publicación del Consejo Nacional de Investigación de los Estados Unidos en el 2011 llamada *Toward Precision Medicine: Building a Knowledge Network for Biomedical Research and a New Taxonomy of Disease* [1]. La misma enuncia la necesidad de crear una nueva forma de clasificación de las enfermedades basada en sus mecanismos de acción y no en sus efectos (síntomas). Ésta sería posible gracias a la capacidad de obtener datos genómicos y corresponderlos con datos de registros clínicos electrónicos para obtener una clasificación de fenotipos muy poderosa.

Esta publicación define *medicina de precisión* como “[...] clasificar individuos en subpoblaciones que difieren en su susceptibilidad a una enfermedad en particular o a su respuesta a un tratamiento específico. Intervenciones preventivas o terapéuticas pueden entonces concentrarse en aquellos quienes se beneficiarán, evitando gastos y efectos colaterales indeseables a los que no.” También define al término *medicina personalizada* como equivalente, pero aclara la preferencia del uso del anterior debido a que el adjetivo *personalizado* pareciera dar alusión a que se generarían nuevos tratamientos para cada individuo, cuando en realidad es para estratos de pacientes.

El potencial de este avance en la medicina es enorme, y sigue dando pasos importantes [2], en particular en la oncología [3] y la **farmacogenómica** [4]. Un pilar esencial para la comprensión molecular de las enfermedades es el Genoma Humano, que la comunidad científica actualmente define como la **secuencia de referencia**, que fue inicialmente armada por el Proyecto Genoma Humano [5] y actualizado periódicamente por el *Reference Genome Consortium*. La versión más reciente del genoma de referencia es GRCh38, tiene 3.257 millones de bases, 20.376 genes codificantes y 22.305 no codificantes. Los genes varían en su longitud desde 8 pares de bases, para un ARN de transferencia y tienen hasta 2,47 millones de pares de bases para el caso de CNTNAP2. Pueden tener un exón y hasta 363 exones, en el caso de la titina. Queda claro que trabajando con el genoma se manejan números astronómicos y la complejidad es muy alta. Al finalizar el Proyecto Genoma Humano, que fue una proeza en sí, restaba una tarea aún más desafiante: el interpretar qué es lo que dice la secuencia de referencia del genoma humano.

Todo este proceso de descubrimiento tiene todavía que enfrentar muchas limitaciones, como la falta de consenso sobre estándares de procedimientos, niveles de calidad mínimos en secuenciación, optimización de algoritmos y tratamiento de datos [6], por sólo mencionar algunos. También existen retos tecnológicos para mejorar la secuenciación, que todavía tiene debilidades considerables [7]. No menos importante es lograr definir con mayor exactitud la relación entre variantes genéticas y enfermedades al establecer relaciones de causalidad entre las mismas [8]. Una buena parte de los genes que se reportan como conectados de manera causal a enfermedades monogénicas son afirmaciones correctas, pero aún existen muchas falsas atribuciones de causalidad entre variantes del genoma y enfermedades.

### 1.1.1. Secuenciación masiva en paralelo

El Proyecto Genoma Humano trajo consigo una esperanza de avance en la medicina con la anhelada comprensión genética de las enfermedades [5]. Los avances tecnológicos en secuenciación lograron no solo que crezca la capacidad de lectura sino una reducción dramática en los costos [9]. Esto ha permitido que plataformas comercialmente disponibles aparezcan alrededor del 2008-2009 y comiencen a estar presentes en el ámbito clínico.

La primer plataforma comercialmente disponible de **secuenciación masiva en paralelo** (*Next Generation Sequencing* o NGS) fue **Roche 454**, desarrollado por 454

Life Sciences, en 2005. Después aparecieron dos grandes empresas cada uno con una tecnología ligeramente distinta, que son **Illumina** y **Ion Torrent**. Actualmente Illumina domina el mercado, y el 70 % de sus equipos vendidos son modelo HiSeq o MiniSeq. Últimamente hay nuevos desarrollos de secuenciación llamados de tercera generación, que consiste en lectura de una sola molécula. Entre éstos actualmente hay tres empresas desarrolladoras: Helios(provisto por SeqLL) , Pacific Biosciences y Oxford Nanopore [10].

### 1.1.2. Tecnologías de secuenciación NGS

La aproximación a secuenciación de lectura corta (*short-read sequencing*) es la predominante en las plataformas NGS actuales, y se basa en procesar en paralelo muestras de ácidos nucleicos cortas, que son parte de una muestra mayor que fue subdividida y amplificada. Existen dos grandes categorías de secuenciación de lectura corta, que son la **secuenciación por ligación** y la **secuenciación por síntesis**.

La secuenciación por ligación consiste en la hibridación de una sola cadena de ADN a un *primer* en una posición conocida y posteriormente la ligación de cadenas sonda complementarias con una o dos bases convencionales. El resto de las bases son degeneradas, de largo conocido y están adheridas a un fluoróforo utilizado como señalador en el momento de la ligación. Estas sondas se van ligando y se va conociendo la base complementaria en intervalos constantes. Las plataformas que utilizan esta tecnología son SOLiD y Complete Genomics [11]. Esta tecnología, debido a su complejidad, altos costos y requerimiento de mano de obra sumamente calificada cada vez es menos utilizada y reemplazada por la secuenciación por síntesis.

La secuenciación por síntesis, a diferencia de la anterior, se basa en usar una polimerasa para sintetizar la cadena complementaria de la muestra, con cada base que se añade. Dicha adición emite algún tipo de señal diferencial para identificar de qué nucleótido se trata y así conocer la secuencia paso a paso. La secuenciación por síntesis también se puede subdividir en dos categorías: terminación cíclica reversible (CRT - *cyclic reversible termination*) y adición de un solo nucleótido (SNA - *single nucleotide addition*).

En CRT, se utilizan bases con un terminador que bloquea el avance de la polimerización. Aplicando un desbloqueante se puede proseguir con el proceso. Esto permite que en cada ciclo se coloquen los cuatro tipos de deoxinucleótidos simultáneamente.

Las plataformas Illumina identifican el nucleótido ya que cada uno está ligado a un fluoróforo de un color que lo identifica y los equipos tienen cuatro canales de detección correspondientes. En el caso de los equipos NextSeq y MiniSeq (Illumina), se tiene un sistema de dos fluoróforos.

Para el caso de SNA, los deoxinucleótidos no tienen un bloqueador terminal y tampoco una emisión particular para cada base. Al emitir todos una misma señal, estos sistemas deben agregar solo un tipo de base nucleotídica en cada ciclo. Un problema en estos casos son las regiones homopoliméricas, en las cuales la polimerasa avanza sin detenerse y provoca una señal no necesariamente proporcionada al número de bases añadidas. La primera plataforma de NGS, 454, utiliza esta metodología. En este caso en particular, al añadirse cada nucleótido se desencadena una cascada enzimática que culmina en una señal bioluminiscente que es detectada por una cámara CCD [12].

Ion Torrent, otro tipo de secuenciación SNA, detecta los iones de hidrógeno resultantes de cada incorporación de un deoxiribonucleótido a la cadena. Es, en efecto, un medidor de pH que es transducido por un transistor ISFET (ion-sensitive field-effect transistor). Como en regiones homopoliméricas se puede incorporar más de un nucleótido, el sensor detecta un menor pH, pero de manera imperfectamente proporcional y limitada [13].

Existen también plataformas que se consideran de secuenciación de lectura larga, particularmente un tipo de tecnología que ha estado creciendo es la de secuenciación de lectura larga de una sola molécula (*single molecule long-read sequencing*). Consisten en la detección de una sola cadena de ADN. Dos plataformas se destacan que son el **PacBio** de Pacific Biosciences y **MinION** de Oxford Nanopore Technologies. PacBio, tiene cubetas que tienen volúmenes en el orden de los picolitros, en cuyo fondo se encuentra una polimerasa adherida. Al estar fija, se conoce la posición espacial donde ocurre la reacción de incorporación de nucleótidos con mucha precisión. Cada base emite una señal luminosa y es leída con un detector comprendido por un láser y una cámara [14]. También tiene la particularidad de que sus muestras son circulares, permitiendo la secuenciación múltiple de cada sitio.

Oxford Nanopore tiene una propuesta diferente. Su sistema consiste en una proteína que envuelve al ADN también adherida a una superficie, pero en este caso la molécula de ADN atraviesa la superficie mediante un poro en el cual está adherida la proteína. Esta proteína tiene un sensor de tensión eléctrica y percibe cambios que son particulares para un conjunto de bases determinado [15].

Todas estas plataformas tienen sus fortalezas y debilidades así como distintos costos asociados. En general, existe una relación inversa entre inversión inicial en equipos y su posterior gasto en insumos. Es por ello que los equipos más costosos poseen insumos más económicos, mientras que los equipos menos costosos luego requieren de un alto gasto en insumos. Aún así, el avance de estas tecnologías reduce los costos en términos generales y permite una mayor estandarización y automatización de los procedimientos, facilitando su uso en el diagnóstico clínico.

### **Ion Torrent**

En el laboratorio de acogida, el Laboratorio de Secuenciación del Hospital Italiano, se trabaja con la plataforma Ion Torrent *Personal Genome Machine*. El **ARN** puede encontrarse en genomas virales, como es el caso del virus de Hepatitis C, y por lo tanto es el componente del que se parte. Para su secuenciación se trabaja con ADN complementario (cDNA) a través de una **transcripción inversa** con el uso de la **transcriptasa inversa**, una enzima encontrada en varios tipos de virus.

Por último la muestra pasa a amplificarse por reacción en cadena de la polimerasa (PCR), esencial cuando se desea secuenciar porciones específicas de un genoma. Dado que este ADN suele ser más largo que la longitud óptima para el equipo, se debe fragmentar aún más por sonicación, que consiste en el uso de ondas de sonido de alta frecuencia para fragmentar moléculas de ADN [16].

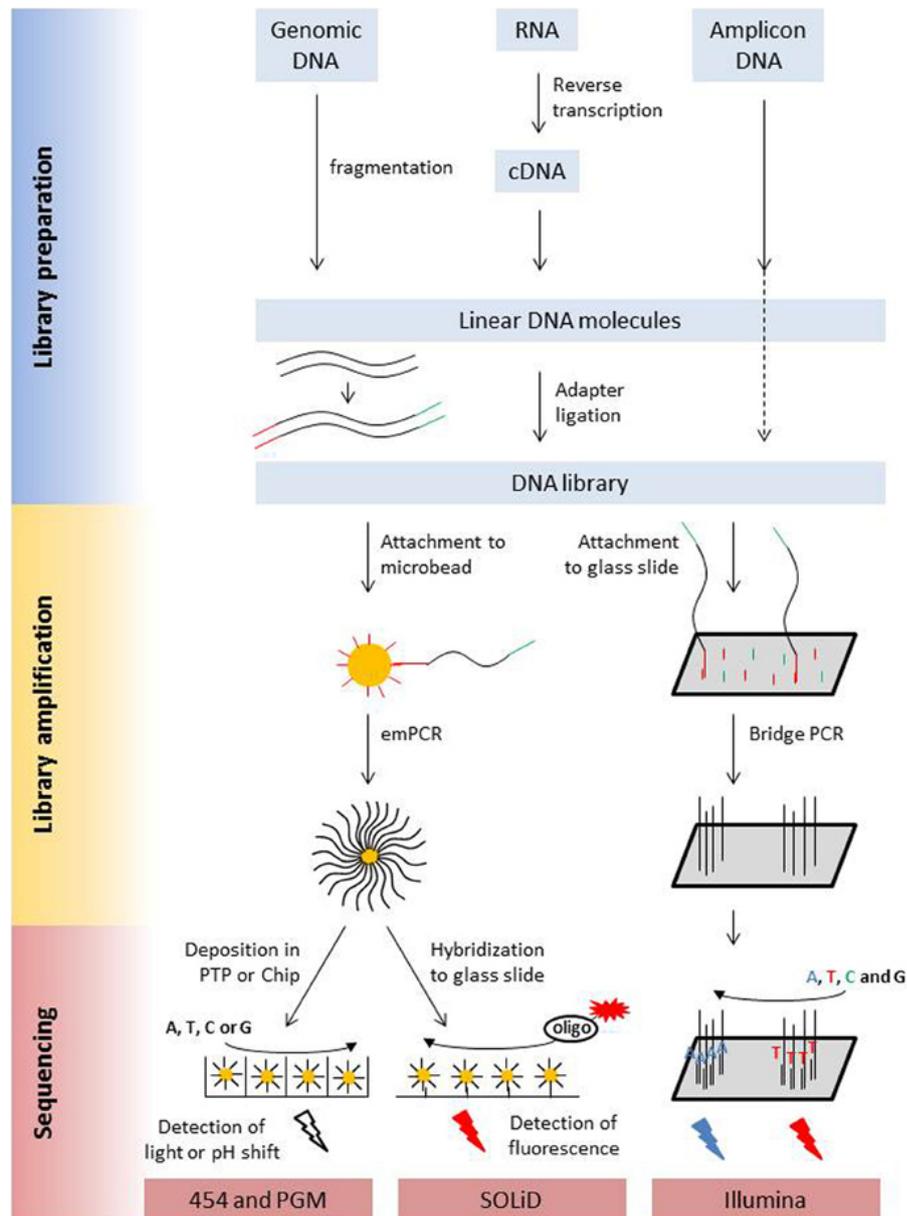


Figura 1.1.: Procedimientos de preparación de muestras para plataformas NGS. Knief 2014

El siguiente paso en la preparación de muestras consiste en ligar adaptadores en los extremos de los pedazos lineales de ADN. Estos adaptadores son cadenas cortas de ADN de secuencia conocida y que sirven para ser adheridos a una cadena complementaria que está fijada a una superficie. De esta forma todas las muestras de ADN pueden ser fijadas para comenzar el proceso de secuenciación. A este conjunto de moléculas de ADN de la muestra se le llama **biblioteca**.

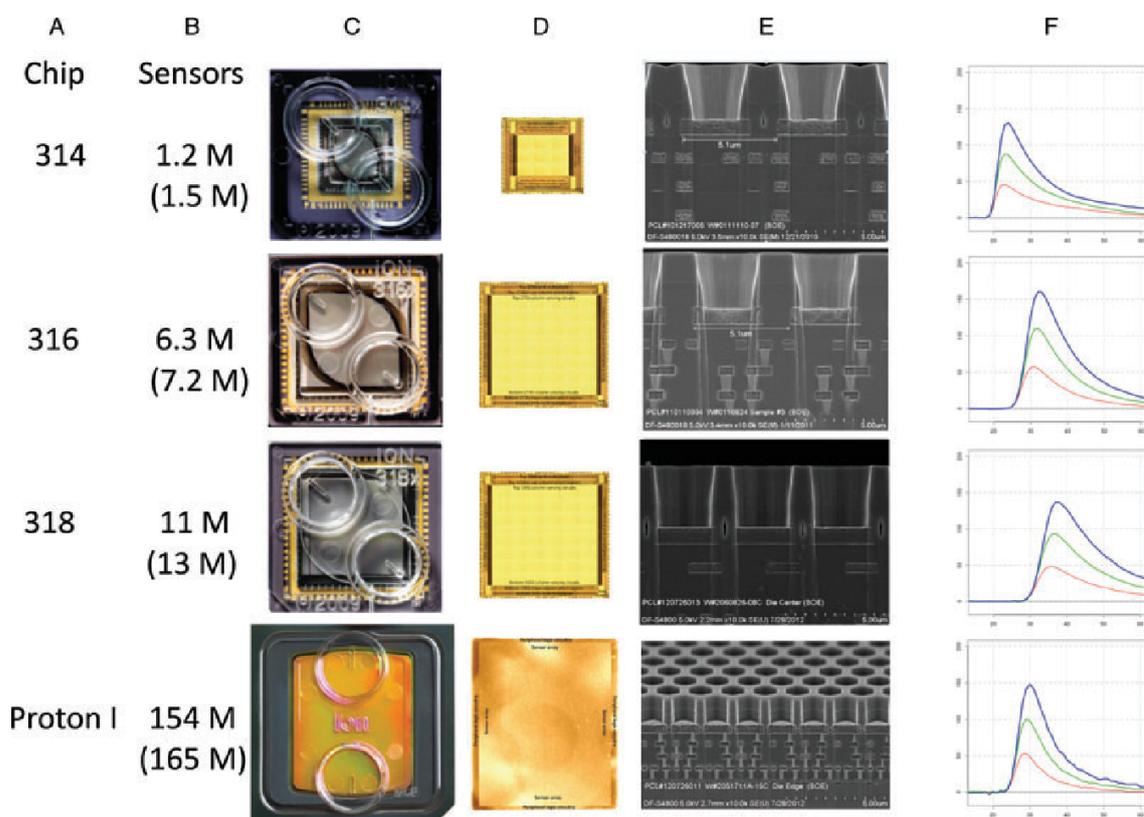
Para aumentar la intensidad de la señal para el proceso de secuenciación, se requiere la amplificación por PCR de las moléculas de la biblioteca. La amplificación tiene que ocurrir espacialmente separada para los fragmentos individuales de la biblioteca.

En el caso de Ion Torrent [17], la separación espacial de los fragmentos individuales se logra a través de microesferas. Las microesferas son de metal, que tienen adherida una gran cantidad de cadenas complementarias a la secuencia de los adaptadores. En estas microesferas cada esfera obtiene, idealmente, una sola molécula de la biblioteca. Lo que se busca es que la esfera, que esta llena de adaptadores, termine con cada adaptador ligado a una de las cadenas del pedazo de biblioteca que se pegó originalmente. Esto se logra a través de una PCR en emulsión (emPCR), en la cual las bolitas se separan espacialmente entre sí en gotitas de agua individuales en una emulsión agua-aceite. Las esferas después se pasan al chip semiconductor donde cada una cae en una micro cubeta llamada *microwell*. Se puede apreciar gráficamente en el lado izquierdo de la figura 1.1, PGM corresponde a un modelo de Ion Torrent.

Los fragmentos de ADN en cada microesfera son de una sola cadena y se les carga una polimerasa. La secuenciación se logra al proveer por flujo, soluciones de un solo nucleótido de manera secuencial. Después un sensor de pH (llamado pHFET) monitorea la señal de incorporación de éstos a la cadena por la polimerasa, ya que una incorporación de un nucleótido por la polimerasa resulta en la liberación de un ion H<sup>+</sup>, lo que disminuye el pH en el *microwell*. Se agregan cuatro soluciones: A,C,T y G para las pruebas de incorporación. Si la solución de un nucleótido no resulta en una incorporación, no se sensa un cambio en pH y no se registra un cambio de señal. En caso de que haya incorporación el pH disminuye y sí se registra un cambio. Entre cada flujo de solución de un solo nucleótido, se hace un lavado para retirar todos los nucleótidos de la anterior solución. Se continua siguiendo un patrón de soluciones, por ejemplo A,C,T,G,A,C,T,G de manera cíclica hasta terminar con las cadenas.

La capacidad de secuenciación de cada chip está determinada por la cantidad de sensores en el mismo. Cada sensor corresponde a una sola cubeta que es capaz de contener una sola esfera, que corresponde a un solo fragmento de ADN de la biblioteca preparada. La progresión en el diseño de chips ha logrado de que se haya avanzado en la cantidad de sensores en cada uno. En el modelo PGM de Ion Torrent, los chips 314, 316 y 318 contienen 1.2, 6.3 y 11.3 millones de *microwells*, respectivamente. El modelo Proton, más potente que el anterior, tiene chips con 165 (Proton I) y 660(Proton II) millones de *microwells*. La figura 1.2 muestra los modelos de chips y sus características principales.

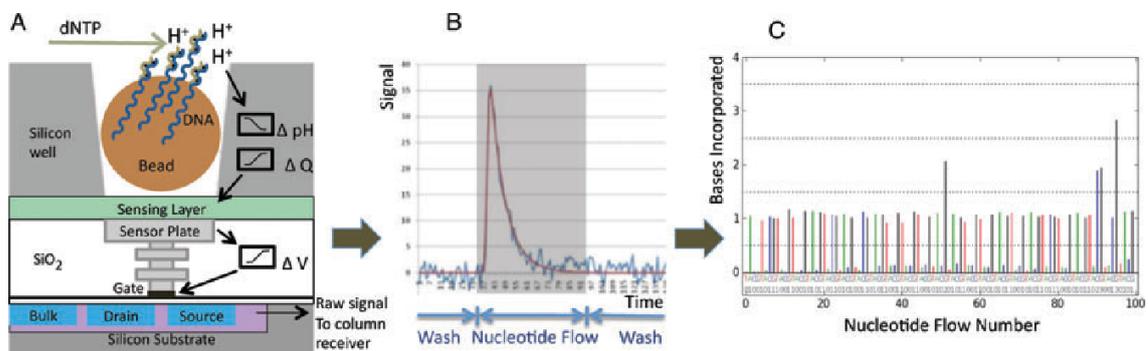
Aumentar el número de sensores permite incluir una mayor cantidad de fragmentos, condicionando así la cantidad de ADN que se podrá secuenciar. Este ADN se puede aprovechar para la extensión genómica que se desea leer o la cantidad de veces que cada base será leída en cada posición. A esto último se le llama **profundidad** (*sequencing depth*), que es el número promedio de *reads* que representa un nucleótido determinado en la secuencia reconstruida. La profundidad dependerá de la cantidad de *reads* y el largo de las mismas. Por ejemplo, un genoma hipotético de 2000 pares de bases con ocho *reads* de 500 nucleótidos cada uno, tendrá una profundidad de 2x. Esa misma cantidad de *reads*, para un genoma de 4000 pares de bases, tendría una profundidad de 1x. Por el contrario para un genoma de 1000 pares de bases, se obtendría una profundidad de 4x.



**Figura 1.2.:** Escalamiento de Chips. (A) Nombre del Chip. (B) Número de microcubetas y sensores accesibles con el total en paréntesis. (C) Imágen con vista superior del chip. (D) Tamaño relativo del area de sensores en el chip. (E) Secciones transversales vista por microscopía electrónica a través del arreglo de sensores (microcubetas individuales apreciables). (F) La forma de función de sensado con cada Chip. *Merriman et al. 2012*

Con esto termina el proceso físico de secuenciación y comienza el flujo bioinformático. El flujo de análisis primario consiste en la recopilación de datos brutos y

procesamiento de señales, seguido del llamado de bases. El llamado de bases consiste en relacionar la intensidad de pH medido con el nucleótido que fue pasado en dicho momento. Todo esto es procesado en un software configurado a medida y optimizado para el ancho de banda y las cargas computacionales. Los tipos de archivo resultantes de este análisis primario son DAT, WELLS y SFF que contienen las señales adquiridas de más bajo nivel, también la normalización de estas señales y su paso a llamados de base y puntajes de calidad. La figura 1.3 ilustra este procesado inicial de la información.



**Figura 1.3.:** Dispositivo de sensor de pH semiconductor y su funcionamiento (A) Configuración de microesfera en un *microwell* con sensor pHFET.(B) Las señales de pH son muestreadas a una alta frecuencia (curva azul) y luego obteniendo una figura filtrada (curva roja). (C) Las señales de incorporación para cada flujo, mostrando la ausencia de incorporaciones, e incorporaciones de 1, 2 y más bases seguidas. Esta señal es el fundamento para el llamado de bases. *Merriman et al. 2012*

El siguiente paso viene a ser el formato **FASTQ**, que es el primer tipo de dato que muestra los resultados de secuenciación como una secuencia de bases. El formato es simplemente una lista de todos los pedazos de ADN de la muestra que llegaron a ser amplificados por la PCR de emulsión y que quedaron exitosamente en un *microwell*. El formato tiene cuatro líneas por cada tira de ADN leído y el texto está en formato ASCII:

1. Primera línea: es el ID que comienza con una @ y es un identificador de secuencia.
2. Segunda línea: es la secuencia misma en letras.
3. Tercera línea: comienza con + y tiene el mismo ID de la primera línea.
4. Cuarta línea: tiene los puntajes de calidad donde la escala esta representado por el orden de todos los símbolos ASCII.

Se puede decir que este tipo de datos es el resultado de interpretación biológica más crudo que sale del secuenciador.

## Diagnóstico Genético Actual

Existen formas adicionales de hacer diagnósticos genéticos, y aunque tienen menor rendimiento, siguen siendo ampliamente usados y resultando más eficientes en determinadas situaciones.

El *gold standard* de la secuenciación es el método **Sanger**, y fue el método de secuenciación más usado hasta la llegada de NGS. De hecho, fue la técnica que se usó para el Proyecto Genoma Humano. Consiste de realizar una amplificación por PCR normal y a medida que se producen enlongaciones de cadenas se agregan nucleótidos con un bloqueador terminal. Estos detienen la enlongación en una base en particular que además tiene un marcador bioluminiscente. Luego estos fragmentos se separan por electroforesis, resultando el orden de tamaño coincidente con el orden de las bases, las cuales son reconocidas por el color de emisión de la señal [10].

La **PCR cuantitativa** utiliza la reacción de PCR para detectar objetivos de interés. Desarrollada a principios de los 90 [18], se utilizan *primers* específicos para un gen y el objetivo se detecta mediante la incorporación de un colorante específico de ADN bicatenario o mediante la liberación de una sonda TaqMan FRET (transferencia de energía de resonancia de Forster). Es ampliamente utilizado tanto en entornos clínicos como de investigación para genotipificación, análisis de expresión génica y detección de patógenos. Es un procedimiento muy rápido y robusto, característica muy importante para análisis con consecuencias en tratamiento médico [19]. Su alta sensibilidad y especificidad lo convierten en el *gold standard* para la detección clínica de genes con varias pruebas aprobadas por la Administración de Drogas y Alimentos de los Estados Unidos (FDA). El número de objetivos simultáneos que se pueden detectar son tan solo cientos, a diferencia de los miles que se pueden detectar con *microarrays* y NGS.

Los **microarrays** de ADN, que consisten en sondas de ADN monocatenarias se inmovilizan en un sustrato en una ubicación discreta con manchas tan pequeñas como 50  $\mu\text{m}$ . El ADN objetivo se marca con un fluoróforo y se hibrida con el conjunto. La intensidad de la señal se usa para determinar el número de moléculas unidas. Los microarrays se utilizan en muchas aplicaciones. Los *arrays* de polimorfismo nucleótido único (SNP) identifican polimorfismos comunes asociados con la enfermedad y los fenotipos, incluidas las enfermedades cardiovasculares, el cáncer, los patógenos y el origen étnico [20].

Los microarrays siguen siendo ampliamente utilizados en la investigación genómica. Se usan para identificar SNPs a costos muy por debajo de las rutinas NGS. Esto también es cierto para los estudios de expresión, en los que los microarrays miden de forma económica los niveles de expresión de miles de genes [21].

## 1.2. Flujo Bioinformático

En la adopción clínica de la tecnología NGS está implícita la necesidad de que la bioinformática procese y ayude en la interpretación de la cantidad masiva de datos generados por los instrumentos de secuenciación [22]. La bioinformática es una disciplina que desarrolla y aplica herramientas computacionales avanzadas para analizar e interpretar datos biológicos de alta dimensión. El papel del bioinformático y los flujos de trabajo de bioinformática son nuevos en los laboratorios de secuenciación clínica y requieren inversiones considerables en educación, personal y *hardware*, así como plasticidad en los procesos y partes involucradas en las pruebas.

El análisis bioinformático basado en NGS está diseñado para convertir señales en datos, datos en información interpretable e información en conocimiento de utilidad clínica. Este análisis es representado como análisis primario, secundario y terciario [23](ver figura 1.4). En resumen, el análisis primario consiste en procesar señales crudas del instrumento de secuenciación, como fue mencionado en el caso de Ion Torrent, y convertirlos en datos de bases de nucleótidos. El análisis secundario consiste en la alineación de la secuencia de bases leídas con una secuencia de referencia seguidas por una detección de variantes. Por último, el análisis terciario es el que evalúa el contexto de la información generada durante un experimento NGS, evaluando el perfil genómico específico de una muestra tomando en cuenta descripciones más profundas de la situación.

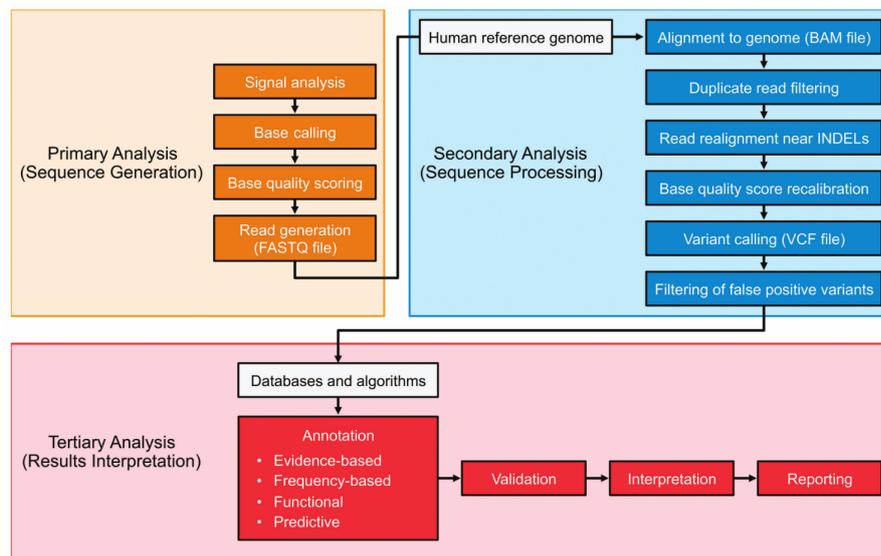


Figura 1.4.: Etapas del análisis bioinformático *Oliver et al. 2015*

### 1.2.1. Alineamiento

El archivo FASTQ es una lista de los fragmentos de la muestra de ADN, pero lo que se quiere es la muestra antes de haberla fragmentado. Es por eso que, conociendo el genoma de referencia del organismo que estamos analizando, podemos alinear estos fragmentos y encontrar a qué parte del genoma pertenecen. Así, se pueden encajar todos los fragmentos y reestructurar la porción entera de ADN buscado.

Existen diversas herramientas de alineamiento, basado en diversos algoritmos, cada uno con sus propias fortalezas y debilidades [24]. El Ion Torrent Suite tiene un alineador llamado Torrent Mapping Alignment Program (TMAP), que está basado en la combinación de tres algoritmos: BWA, SSAHA y Super-maximal Exact Matching.

En cuanto a herramientas *open-source*, uno de los algoritmos más ampliamente usados es el BWA (Burrows-Wheeler Aligner), que se basa en la **transformada de Burrows-Wheeler**. El uso en este proyecto fue BWA-MEM [25], planteado por Li H. en el 2013, del cual se ha reportado un alto porcentaje de lecturas propiamente alineadas [24].

Necesita dos archivos de entrada, un genoma de referencia y un archivo de entrada FASTQ. El resultado es un archivo **SAM**, que significa *Sequence Alignment Map*, y este archivo contiene los *reads* (fragmentos leídos) y su posición en el genoma resultante del alineamiento. También contiene el puntaje de mapeo, que se basa en un sistema de

calificación para evaluar cuantos cambios en la secuencia del genoma fueron necesarios para que encaje un *read* en esa posición.

Los archivos SAM generalmente son archivos muy pesados, y es por eso que en la práctica común se pasan a un registro binario, y el archivo resultante es un archivo BAM. El archivo BAM se puede obtener con Samtools, un paquete de programas para manipular archivos SAM [26].

Samtools también provee herramientas para los siguientes pasos en el flujo bioinformático, la función *sort* ordena los *reads* de izquierda a derecha con respecto a sus coordenadas en el genoma de referencia. La función *index* crea un archivo BAI, que es un archivo independiente conteniendo el índice de posiciones de los *reads* del BAM.

### 1.2.2. Variant Calling

A la hora de secuenciar una muestra, lo que más nos interesa es cómo éste varía con respecto al genoma de referencia, y esto ocurre sólo en algunas posiciones en particular. Es por eso que el siguiente paso se llama *variant calling* o llamado de variantes y consiste en crear un programa que encuentre (o llame) a estas variantes que tienen tanto valor. A estos programas se les llama *variant callers*.

Este proceso no es tan fácil como parece, ya que hay que separar lo que son variantes reales en ADN de artefactos o errores que surgen de la preparación de bibliotecas, la secuenciación o en el alineamiento. A lo largo de los años se han desarrollado muchos algoritmos para *variant callers*, la mayoría de código abierto y con publicaciones que respaldan su rendimiento.

A la hora de diseñar un *variant caller*, la estrategia usada va a cambiar mucho dependiendo de qué tipo de variantes se desean encontrar. Existe una clasificación de variantes dependiendo de cuándo ocurren, y éstas son las variantes germinales o somáticas. Las variantes germinales aparecen durante la formación de las células sexuales, y se encuentran en todas las células del organismo que se origine de éste. Las variantes somáticas ocurren después de la fertilización y dependiendo de la etapa de desarrollo embrionario que ocurrieron, la proporción de células del cuerpo que se verá afectada será distinta. En el caso de variantes germinales, se espera que estas aparezcan solo en el 50 % o 100 % de la fracción de las muestras, también llamada frecuencia alélica. Pero para muestras impuras, tumorales, o de poblaciones de microorganismos,

esta fracción puede ser cualquiera. Las variantes de muy baja frecuencia, menores a 5% resultan las más difíciles de diferenciar de artefactos.

También existe una clasificación en cuanto a *cómo* surgen las variantes, y están agrupados en tres categorías, que son: variantes de nucleótido único (SNV, *single nucleotide variant* o **SNP**), inserción o deleción (**indel**) y variantes estructurales (**SV**).

Los *variant callers* por lo tanto son diseñados para especializarse en algún tipo de variante. En este trabajo el enfoque está puesto en las SNPs, particularmente en sustituciones asociadas a resistencia (RAS - *Resistance Associated Substitutions*). Por lo tanto se realizaron pruebas con los siguientes *variant callers*, reportados en literatura [27–32], son de código abierto y preparados para datos NGS:

- Lofreq
- VarScan
- GATK
- Platypus
- FreeBayes

### Lofreq

Lofreq, desarrollado por Wilm et al. [33] basa su criterio de selección con una fuerte dependencia en los puntajes de calidad del llamado de bases del secuenciador y puntajes de mapeo, que son ignorados por muchos otros callers. No está diseñado para una plataforma de secuenciación en particular y se adapta a cambios de cobertura y calidad de secuenciación, lo que permite que sea utilizado en muchos tipos de datos. Está enfocado en variantes de muy baja frecuencia alélica y tiene la fortaleza de detectar variantes por debajo del promedio de puntaje de calidad del llamado de bases. Está programado en C.

## Varscan

Varscan es un programa en Java y tiene funciones separadas para llamar a SNVs o indels [34]. Se basa en criterios estadísticos para su selección. Toma como entrada un archivo distinto al BAM, llamado archivo *pileup* que se obtiene con Samtools.

## GATK

El Genome Analysis Toolkit es una colección de herramientas para línea de comando programado en Java, dentro de las cuales se encuentran dos *variant callers*: HaplotypeCaller y MuTect 2 [35]. Está desarrollado por el Broad Institute. Para esta práctica se utilizó el HaplotypeCaller debido a su mayor versatilidad, ya que MuTect 2 está enfocado sólo a muestras tumorales humanas. La fortaleza del HaplotypeCaller es que hace una verificación adicional por medio de una reacomodación *de novo* local de los *reads* sin tomar en cuenta el genoma de referencia, y así logra deshechar errores de mapeo.

## Platypus

Platypus es un *variant caller* desarrollado por Rimmer et al. [36] y está programado en Python, Cython y C. Su estrategia principal se basa en realineamiento local de los *reads* para corregir errores de mapeo.

## Freebayes

Freebayes es un *variant caller* con capacidad de detectar SNPs, Indels y MNPs (polimorfismos de múltiples nucleótidos) con una aproximación Bayesiana para filtrar [37]. Realiza pruebas flexibilizando las posiciones de alineamiento, que ayuda a detectar errores basados en alineamiento.

### 1.2.3. Formato VCF y Variant Annotation

El formato más usado para registro de variantes genéticas es el Variant Call Format, o **VCF**. El archivo comienza con datos generales de la muestra, información sobre el formato y criterios de filtración usados. Después le sigue una tabla, en la cual cada fila es una variante y las columnas describen características de la variante.

Las características del formato VCF para las variantes son:

1. **#CHROM** - El cromosoma donde se encuentra la variante.
2. **POS** - La posición de la variante en ese cromosoma.
3. **ID** - Identificadores predeterminados.
4. **REF** - La base nucleotídica en el genoma de referencia.
5. **ALT** - La base nucleotídica alterna encontrada en la muestra.
6. **QUAL** - Calificación de calidad para la detección de esta variante.
7. **FILTER** - Aclara qué filtros del *variant caller* fueron pasados, en caso de pasar todos dice "PASS".
8. **INFO** - Información adicional, que usualmente son datos de frecuencia alélica (AF), profundidad de secuenciación (DP), si está registrado en la base de datos dbSNP (DB), calidad de mapeo (MQ) y otros detalles más.

El último paso en el procesamiento de datos de secuenciación es la anotación de variantes. Consiste en asociar las variantes y particularidades genéticas encontradas con sus efectos en el fenotipo. Para esto existen muchas bases de datos que tienen anotadas las variantes responsables de efectos en particular, también se puede analizar qué efectos tienen las variantes sobre los aminoácidos de la proteína en cuestión y cómo ésta afecta al organismo.

En este trabajo, este último procesamiento de datos se realizó programando con el lenguaje R, el cual es usado ampliamente debido a la facilidad que otorga para trabajar con bases de datos y herramientas estadísticas. En particular se usó un paquete del repositorio Bioconductor llamado Variant Annotation el cual tiene múltiples herramientas para trabajar cómodamente con archivos VCF. Usando la función *vcfread*, se pasan los contenidos del archivo a formato *dataframe*. Extrayendo las columnas

deseadas, en este caso la columna "AF", que corresponde a las frecuencias alélicas, y "pos", que corresponde a posiciones.

### 1.3. Virus de Hepatitis C

El virus de la Hepatitis C (VHC) causa la **Hepatitis C** y se estima que aproximadamente 62 a 89 millones de personas la padecen, que sería entre 1,2 % a 1,7 % de la población mundial [38]. La forma de transmisión es a través del contacto con la sangre o por transmisión sexual. La enfermedad afecta principalmente al hígado, y aunque el 80 % de los afectados inicialmente no presenta síntomas, la función hepática va siendo deteriorada con los años. Las infecciones crónicas pueden terminar en consecuencias más graves como cirrosis o cancer hepático.

En Argentina, el Ministerio de Salud en su informe "Las Hepatitis Virales en la Argentina (2014)", reporta que en el 2012 había una tasa de 0,83 casos por cada 100.000 habitantes y que ha venido bajando consistentemente desde el 2005 [39]. También informa que de los 7 genotipos del virus, predominaba el genotipo 1.

#### 1.3.1. Genoma y Mecanismo de Acción Viral

El virus de la Hepatitis C es un virus ARN de la familia *Flaviviridae*, en los cuales se encuentran también el dengue y la fiebre amarilla. En la actualidad el virus se clasifica en 7 genotipos, siendo el genotipo 1 el más abundante en la Argentina y en los pacientes estudiados en el Hospital Italiano. El genotipo 1 se clasifica adicionalmente en dos subtipos, el subtipo a y b, de los cuales el predominante es el b por lo que resulta de mayor interés a nivel clínico.

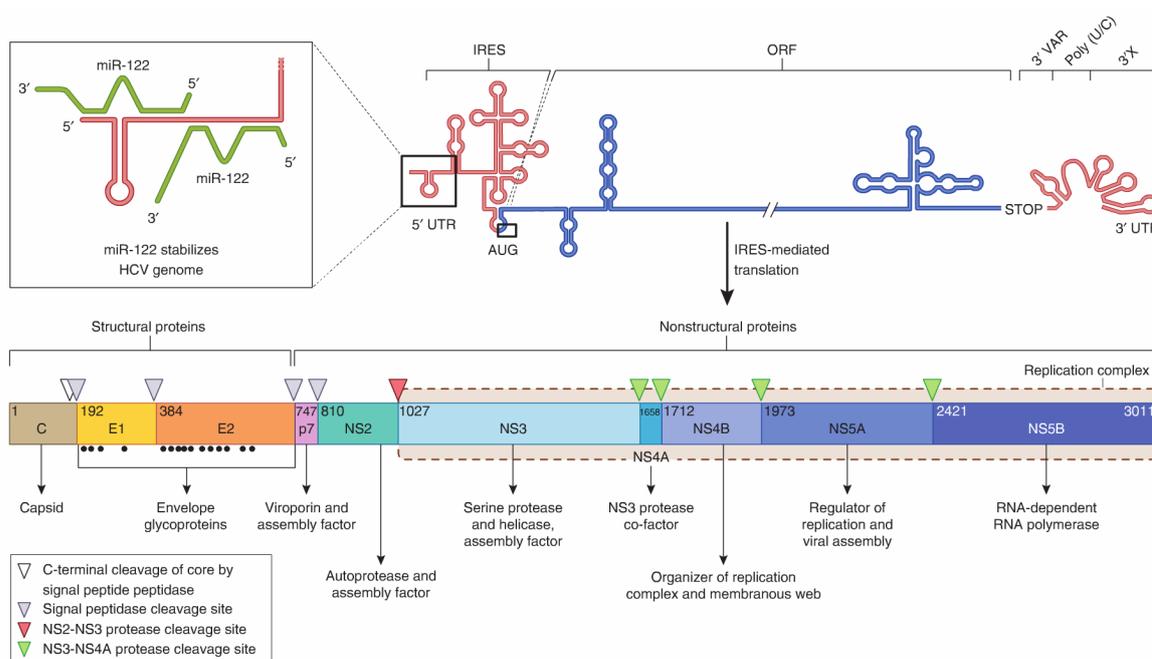
El genoma del virus tiene 9,6 kilobases, es de una sola cadena con polaridad positiva y tiene un solo marco de lectura abierto que resulta en una sola poliproteína directamente traducida por un ribosoma [40]. Su genoma también posee regiones no traducidas (UTR) en ambos extremos (ver Figura 1.5). La poliproteína resultante se traduce en los ribosomas del retículo endoplásmico y se subdivide en diez proteínas, que se separan en dos grupos: proteínas estructurales y no-estructurales.

El ciclo de vida del virus, apreciado en la Figura 1.6, consiste de cinco fases [41]:

1. Circulación de partícula viral.

2. Entrada a la célula huésped e interacción con el receptor.
3. Traducción y procesamiento de la poliproteína
4. Replicación de ARN viral.
5. Ensamblaje y morfogénesis del virión.

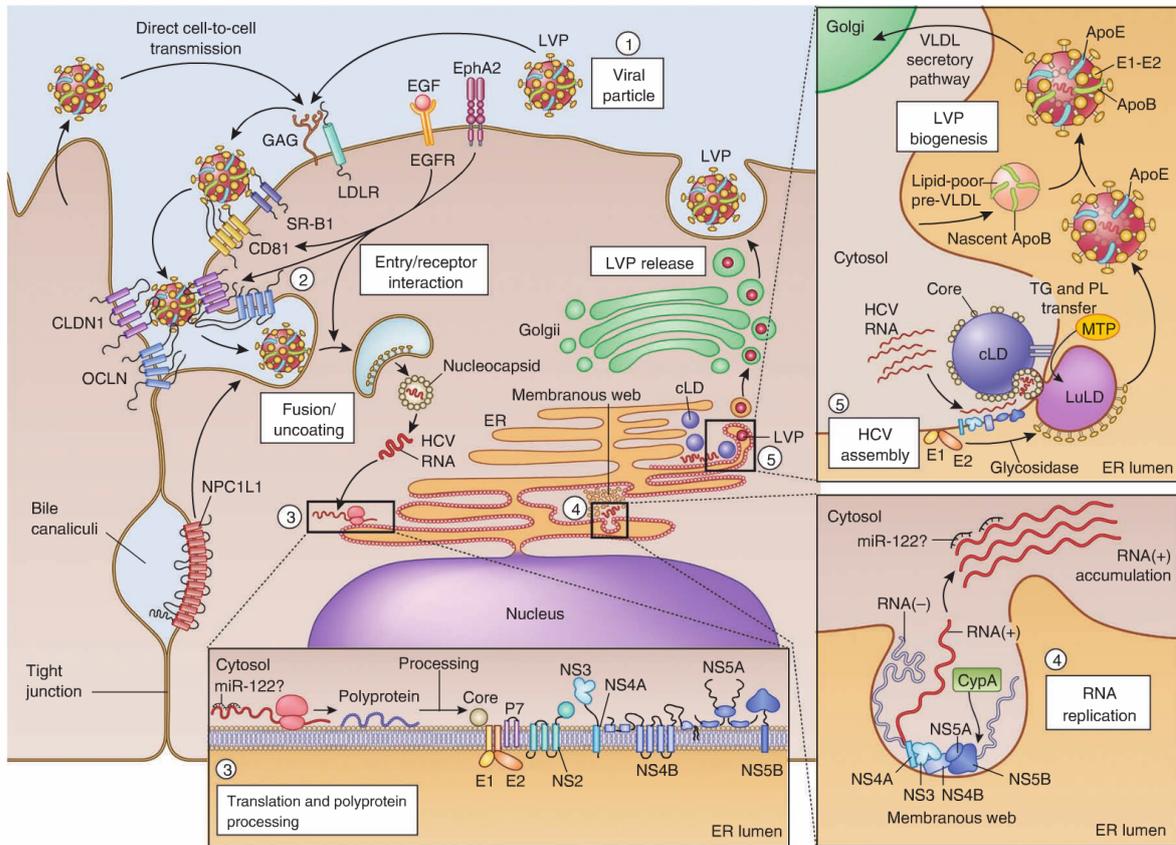
El proceso de traducción y posterior replicación del ARN viral ocurre en la membrana del retículo endoplásmico de la célula huésped. Aquí es donde las proteínas no-estructurales, que son en su mayoría de transmembrana, se encargan de la replicación y de utilizar esta misma membrana para asistir en la liberación de las nuevas partículas virales.



**Figura 1.5.:** Genoma del VHC (arriba) y procesamiento de poliproteína (abajo). Los triángulos representan puntos de corte peptídico, los puntos negros sitios de glicosilación de proteínas. *Scheel et al. 2013*

Las proteínas estructurales son C (cápside), E1 y E2. Las proteínas no-estructurales son p7, NS2, NS3, NS4A, NS4B, NS5A y NS5B.

Las proteínas estructurales participan en el proceso de formación de partículas virales, siendo C el componente del nucleocápside que envuelve directamente el material genético y las proteínas E1 y E2 formando la envoltura.



Points of intervention in the HCV life cycle

- ① The viral particle (neutralizing antibodies, virocidal peptides)
- ② Entry and receptor interaction (antibodies and small molecules targeting receptors, kinase inhibitors)
- ③ Translation and polyprotein processing (NS3-NS4A protease inhibitors)
- ④ HCV RNA replication (NS5B polymerase and NS5A inhibitors, miR-122 antagonists, cyclophilin inhibitors, statins, PI4KIII $\alpha$  inhibitors)
- ⑤ Assembly and virion morphogenesis (NS5A inhibitors, DGAT1 inhibitors, glycosidase inhibitors, MTP inhibitors)

**Figura 1.6.:** Ciclo de vida del VHC: 1.Circulación de partícula viral. 2. Entrada a la célula huésped e interacción con el receptor. 3. Traducción y procesamiento de la poliproteína. 4. Replicación de ARN viral. 5. Ensamblaje y morfogénesis del virión. *Scheel et al. 2013*

Pasando a las proteínas no estructurales [42], p7 es un canal de iónico transmembrana que contribuye en el ensamblaje de la partícula. NS2 también es una proteína transmembrana que es hidrofóbica que actúa en conjunto con el extremo de NS3 para funcionar como una autoproteasa que procesa la unión entre NS2 y NS3. NS3 tiene tres funciones, una ya mencionada en conjunto con NS2, otro extremo que es una serina proteasa y actúa sobre C, E1 y E2. Y el tercer extremo que es una helicasa y contribuye en el proceso de replicación del ARN. NS4A forma otro complejo con NS3 que cortan proteínas de señalización antiviral de la célula huésped, también cortan a las proteínas NS3, NS4A, NS4B, NS5A y NS5B. NS4B se encarga de inducir la producción de esférulas a partir de la membrana del retículo endoplásmico. NS5A participa tanto en la

replicación de ARN en conjunto con NS5B y en la formación de las partículas virales. NS5B es una ARN-polimerasa-ARN-dependiente y es la que directamente produce el copiado de ARN.

### 1.3.2. Tratamientos

El tratamiento tradicional a la Hepatitis C es la combinación de interferon pegilado alfa y ribavirina, por un lapso de 24 a 48 semanas [38]. La obtención de respuesta virológica sostenida (SVR), es de 40 %-50 %. Este tratamiento colabora al sistema inmune del paciente a luchar contra el virus, pero en el 2011, se aprobó el uso de un nuevo medicamento que proponía una estrategia diferente. Esta estrategia consiste en atacar directamente al virus y viene de la mano de los **antivirales de acción directa** (*Direct Acting Antivirals* - DAA's) [43]. Los primeros aprobados fueron boceprevir y telaprevir, con una tasa de SVR de hasta 70 %. Después en 2013 la llegada de simeprevir y sofosbuvir elevaron la tasa a 90 %, proporción que en la actualidad resulta normalmente esperable. Los DAA's se enfocan en interferir la acción de tres funcionalidades específicas del virus: NS3-NS4A, NS5B y NS5A. Los que atacan a NS3-NS4A contienen el sufijo *-previr*, de NS5B el sufijo *-buvir* y los de NS5A el sufijo *-asvir*. El último mencionado, que afecta al NS5A es la categoría más novedosa y también tiene resultados potentes en combinación con otros DAA's [44]. Estos compuestos siempre se dan en combinaciones, algunas particularmente más efectivas para cada genotipo. Estos tratamientos, a medida que son expuestos al virus, provocan que éste desarrolle resistencias. Se discutirán las implicaciones de las resistencias a tratamientos y cómo se puede responder a estas situaciones, que resulta el problema central de este trabajo.

### 1.3.3. Variantes Asociadas a Resistencia

El VHC es capaz de adquirir resistencia a cualquier drogas de acción directa desarrollada, ya que tiene una tasa de replicación acelerada y la enzima que cataliza este proceso, la ARN-replicasa, no posee mecanismos de corrección a errores en su tarea [41]. Esto resulta en grandes números de diversas variantes virales en cada individuo afectado, muchas de las cuales resultan con susceptibilidad reducida a los antivirales [45]. Las variantes resistentes eventualmente forman, por selección natural, poblaciones resistentes que logran ser inmunes a un tratamiento antiviral en particular [46]. Muchas veces, la sustitución de un solo nucleótido puede conferir resistencia a

un antiviral. Y no solo a una droga específica, sino que puede resultar en resistencias cruzadas que afectan a toda una clase de antivirales [46]. Se han identificado las mutaciones de resistencia que aparecen con mayor frecuencia [43,47]. De esta manera, se pueden establecer múltiples relaciones entre mutaciones específicas y su resistencia a distintas drogas. Con una documentación de las mutaciones pertinentes, es posible identificar con sencillas herramientas bioinformáticas si un genoma en particular es resistente o no [48]. Sin embargo, en una muestra poblacional de virus la situación cambia, en un paciente se tienen muchas variantes de los distintos especímenes virales y tan solo una proporción es resistente.

La secuenciación NGS devuelve resultados de una muestra que contiene múltiples genomas mezclados y, por lo tanto, distintas secuencias con una frecuencia de aparición particular. Cuando el porcentaje de aparición de la mutación es muy bajo (generalmente menor al 5%), se la denomina “mutación de baja frecuencia” [49]. Éstas resultan más difíciles de identificar, y a menor frecuencia resulta más desafiante lograr diferenciarlas de errores de secuenciación. En la situación previa al tratamiento antiviral, las poblaciones de virus resistentes son generalmente muy chicas, ya que no ha habido una selección natural en favor de estas variaciones. Pero justamente resulta conveniente encontrarlas cuando son menores, ya que conforme avance el tratamiento estas poblaciones seguirán creciendo considerablemente, haciendo menos efectivo el tratamiento y aumentando las probabilidades de recaída.

Por lo tanto, es inobjetable la extrema importancia de encontrar desde el comienzo del tratamiento toda **mutación de baja frecuencia** que condicione la respuesta del paciente, y para hacerlo es necesario emplear herramientas bioinformáticas [49–51], tal y como fueron explicadas en la sección 1.2. De esta manera, con una detección efectiva, se puede predecir la reacción de la población viral a distintas drogas y se puede escoger el **tratamiento dirigido** más efectivo para cada estrato de pacientes [52,53], ayudando al médico en la toma de decisión terapéutica.

## 1.4. Evaluando Variant Callers en el Virus de Hepatitis C

Como fue mencionado anteriormente, existen diversos trabajos en los cuales se comparan los rendimientos de *variant callers* aplicados a muestras humanas [27–32].

Sin embargo, las condiciones en muestras virales son muy distintas y aparecen nuevos desafíos a tener en cuenta. Se han desarrollado múltiples herramientas, tanto para analizar variantes en muestras virales [49,50,54] así como en escenarios más complejos como las estimaciones locales de diversidad de cuasiespecies y reconstrucciones globales de haplotipos virales [55].

Sin embargo, no se encontró ninguna evaluación de los rendimientos de *variant callers* de código abierto aplicados a muestras del virus de Hepatitis C. En las evaluaciones de datos de secuenciación de VHC se utiliza un solo *variant caller* o se tiende a confeccionar uno calibrado al *set* de datos de esa investigación [48,49,56]. Es por eso que este trabajo propone evaluar el rendimiento de distintos *variant callers* para la identificación de variantes asociadas a resistencia, particularmente en frecuencias alélicas bajas del virus de Hepatitis C. Los resultados de tal evaluación permitirían orientar al analista o bioinformático, en la utilización de una o varias herramientas de *variant calling*, ya diseñadas y optimizadas y con nociones de sus fortalezas y debilidades.

## 2. Objetivos

### 2.1. Objetivo General

El objetivo general consiste en implementar herramientas bioinformáticas para la automatización del proceso de identificación de variantes genéticas de baja frecuencia en el contexto de resistencia a fármacos del virus de la Hepatitis C.

### 2.2. Objetivos Específicos

1. Evaluación de herramientas bioinformáticas de código abierto:
  - a) Prueba y Selección de los filtros de llamado de variantes (variant callers) con mejor rendimiento reportados en la bibliografía para mutaciones de baja frecuencia.
  - b) Prueba de herramientas para la manipulación y extracción de data de archivos BAM(Binary Sequence Alignment Map) y VCF (Variant Call Format).
  - c) Análisis de herramientas de traducción de secuencias de ADN y análisis de mutaciones funcionales.
2. Evaluación de las mejores herramientas seleccionadas en el objetivo 1 y acoplamiento a un flujo de análisis de las variantes. Este análisis consiste en un estudio de las posiciones del genoma que se ven más afectadas y sus consecuencias funcionales.
3. Pruebas de análisis de variantes con muestras reales de ARN viral de Hepatitis C de pacientes.

## 3. Materiales y Metodos

### 3.1. Obtención de Datos

En este trabajo se utilizan dos clases de datos, reales y artificiales.

Los datos artificiales se generaron para la totalidad del genoma del virus a fin de obtener una secuencia que funcione como *gold standard* para la detección de variantes. Para ello se utilizó una herramienta *open-source* que produce archivos FASTQ a partir de secuencias en formato FASTA. El procedimiento para obtener los datos generados está detallado en la subsección 3.1.1.

Para el análisis con datos reales, se recurrió a la base de datos del National Center for Biotechnology Information (NCBI) en particular a los archivos que forman parte del BioProject PRJNA433160. A diferencia de los datos artificiales, éstos solo corresponden a la secuencia de las regiones que codifican para las proteínas NS3 y NS5B del VHC. Esto se debe a que no se encontró ningún genoma de referencia del virus completo con suficiente profundidad como para poder distinguir de manera confiable las variantes reales de los artefactos de secuenciación. Por lo tanto, se optó por obtener los archivos FASTQ de aquellas regiones, ya que son las de mayor interés por ser los sitios *target* de los antivirales de acción directa (ver sección 1.3.2). En particular, se utilizaron los genomas del genotipo 1, subtipo b por ser el más frecuente en la población argentina (ver sección 1.3.1).

Estos datos son de pacientes del Hospital de Clínicas de la Universidad de Barcelona, y en total ponen a disposición los datos de 16 muestras para cada proteína, obteniéndose un total de 32 archivos FASTQ que fueron utilizados en su totalidad. La secuenciación fue realizada en la plataforma 454 GS Junior. Estos datos tiene una profundidad de muy poca variabilidad en torno a los 8000x. En el apéndice se encuentran listados las muestras específicas que fueron seleccionadas del Bioproject en cuestión.

### 3.1.1. Obtención de datos artificiales con Artificial Fastq Generator

Artificial Fastq Generator es una herramienta de código abierto, programada en java. Esta herramienta crea archivos FASTQ artificiales a partir de una secuencia en formato FASTA [57]. Debido a que estos archivos fastq son formados a partir de una secuencia conocida, vienen a ser un gold standard para *variant calling*. Es por eso que esta herramienta es crucial para evaluar los *variant callers* en este estudio.

La herramienta tiene múltiples opciones de funcionamiento, de las cuales algunas fueron más relevantes para el análisis en cuestión. A continuación se muestran las opciones utilizadas:

- **F1/F2** - Permite ingresar dos archivos reales FASTQ para utilizar los *Quality Scores* de éstos
- **S** - Identificador de comienzo de secuencia
- **SE** - Simular errores aleatorios en los *Quality Scores*
- **CMP** - El coverage pico promedio para una región
- **CSD** - Desvío estándar de la profundidad dividido por el promedio
- **RL** - Largo de los *reads*
- **TLM** - Promedio del tamaño de los templates
- **TLSD** - Desvío estándar de los templates

Ejemplo de uso por línea de comando:

```
$ java -jar ArtificialFastqGenerator.jar -R ref.fasta -O output.fastq  
-S ">sequence_id" -SE true -CMP 1000 -CSD 0.01 -RL 100
```

Si se usa el genoma de referencia como archivo FASTA de entrada, simplemente se obtendrá una muestra de secuenciación simulada con todas las bases en cada *read* idénticas a las de la referencia. Para poder insertar variantes, se procedió a alterar el FASTA del genoma de referencia cambiando las bases en la posición deseada. Para poder simular la fracción alélica (AF) de estas mutaciones, se generaron dos FASTQ simulados, una proporción usando el genoma de referencia y una segunda proporción usando el genoma con las variantes insertadas. Luego se fusionaron ambos archivos

con la función *cat* desde terminal, obteniendo así un solo archivo con proporciones poblacionales específicas de las variantes insertadas.

Por ejemplo, si se quiere una muestra de profundidad 1000x con una variante *v* y se desea que tenga una AF de 0.2, se generan un archivo FASTQ a partir del FASTA de referencia con una profundidad promedio de 800x, y luego un segundo archivo FASTQ a partir del FASTA de referencia con la variante *v* insertada con una profundidad promedio de 200x. Al fusionar estos dos archivos se obtiene la muestra con las características deseadas.

En este trabajo se generaron FASTQ artificiales con un pico de profundidad promedio de 1000x y un desvío estándar de 10x, un tamaño de reads de 100 bases y con generación aleatoria de errores de lectura, que correspondían a una variante insertada de profundidad 1x. Se escogió un desvío muy pequeño ya que se deseaba que la variabilidad en la profundidad no afecte el cálculo de la AF de los *variant callers* en el proceso de evaluación.

## 3.2. Procesamiento Bioinformático de los Datos

### 3.2.1. Alineamiento

Todas las muestras FASTQ obtenidas, artificiales y reales, fueron alineadas con la herramienta BWA-MEM, Los archivos binarios BAM fueron generados usando Samtools *view* y los índices fueron creados usando Samtools *index*.

El alineador se procesa por línea de comando:

```
$ bwa mem ref.fasta reads.fastq > aln.sam
```

Generación del BAM:

```
$ samtools view -b aln.sam -o aln.bam
```

Para crear el archivo BAI:

```
$ samtools index aln.bam >aln.bai
```

### 3.2.2. Variant Callers

Los *variant callers* empleados en este trabajo fueron seleccionados por tener características que los resaltan en la evaluación llevada a cabo por Sandmann et al. 2017 [27], en la cual realizan un proceso de selección meticuloso y que se alinea a lo que se busca evaluar en el análisis propio. De los ocho *variant callers* originalmente propuestos (GATK, Platypus, VarScan, Lofreq, FreeBayes, SNVer, SAMtools y VarDict) en este trabajo, seis fueron seleccionados y dos (VarDict y SNVer) se dejaron a un lado por su incapacidad de analizar los datos artificiales, que tienen características anómalas comparadas con datos reales. Particularmente se deseó que los callers sean robustos ante la detección en bajas AF, entre las cuales Lofreq y Freebayes se destacan, según reportado en la bibliografía [27]. El uso del *variant caller* de SAMtools está dentro de un *caller* confeccionado por el autor, al que se le llama *naive variant caller* y cuyas características y construcción están detalladas en la sección 3.3.

Para lograr una comparación justa de los *variant callers*, se consideró mantener las opciones recomendadas por defecto. Sin embargo, se seleccionó la opción de detectar exclusivamente SNPs en cada uno de los *callers*. En el caso de Platypus, FreeBayes y GATK, se seleccionaron además opciones para reducir la frecuencia mínima detectada por defecto y se seleccionó una AF=0.001. Sin este cambio sólo encuentran variantes con AF que son demasiado altas para el enfoque de esta evaluación. Se escoge una frecuencia alélica de uno en mil, debido a que en los datos artificiales, que tienen una profundidad de 1000x, esta es la proporción mínima de los errores aleatorios simulados por el software. En el caso de los datos reales, que tienen una profundidad de 8000x, las frecuencias menores a 0.001 también resultan demasiado chicas para el interés en cuestión y con muy alta probabilidad de ser errores de secuenciación.

Las herramientas escogidas se muestran a continuación con la respectiva sentencia por línea de comando para procesar los archivos BAM.

#### Lofreq

Código por línea de comando para Lofreq:

```
$ lofreq call -f ref.fasta -o sample.vcf sample.bam
```

## Varscan

Código por línea de comando para Varscan:

```
$ samtools mpileup -f ref.fasta sample.bam  
| java -jar VarScan.v2.3.9.jar pileup2snp --output-vcf 1 > sample.vcf
```

## GATK

Código por línea de comando para GATKHaplotypeCaller:

```
$ java -jar GenomeAnalysisTK.jar -T HaplotypeCaller -R ref.fasta  
-I sample.bam -ploidy 1000 -o sample.vcf
```

## Platypus

Código por línea de comando para Platypus:

```
$ Platypus.py callVariants --bamFiles=sample.bam --refFile=hcv1b.fasta  
--genSNPs=1 --genIndels=0 --minVarFreq=0.001 --minFlank=0 --output=sample.vcf
```

## Freebayes

Código por línea de comando para Freebayes:

```
$ freebayes -F 0.01 -X -u -r AJ238799.1:8250-8631 -p 200  
-f hcv1b.fasta sample.bam > sample.vcf
```

### 3.3. Diseño del Naive Variant Caller

Un *naive variant caller* es una herramienta que simplemente registra cualquier cambio de las lecturas versus la referencia, sin discriminar condición alguna. Esta

herramienta tiene una sensibilidad perfecta, ya que no se le escapa ninguna variante, pero también abundan los falsos positivos. En este proyecto se creó un *naive variant caller* combinando la función *mpileup* de Samtools, un programa de código abierto en Perl, llamado “pileup2baseindel” y finalmente un acoplamiento y pos-procesamiento en R con un código de elaboración propia. Este *caller* adicional funciona como un control y parámetro de los límites de sensibilidad y valor predictivo positivo a la hora de comparar los *variant callers*.

El procedimiento es el siguiente:

Primero se convierte el BAM a pileup

```
$ samtools mpileup -E -f hcv1b.fasta sample.bam > sample.pileup
```

Segundo paso consiste en obtener una tabla de texto

```
$ perl pileup2baseindel.pl -i sample.pileup > sample.txt
```

En el siguiente paso el programa en R lo que se hace es leer la tabla de texto y pasarla a formato **dataframe**. Esta tabla contiene los números de cada base leídas en cada posición del archivo BAM. A continuación se obtiene la frecuencia de cada base, y se logra dividiendo el número de lecturas de una base por el número total de bases leídas en esa posición. Sumando el número total de bases en cada posición, ya realizado anteriormente en el cálculo de frecuencia, se obtiene la profundidad, que se añade también como una columna de información.

Posteriormente se eliminan todas las lecturas que son iguales a la referencia en dicha posición, ya que éstas no serían variantes. Por último se imponen dos filtros de seguridad. El primero corresponde imponer una frecuencia mínima de 0.001, por el mismo motivo que el expuesto en la sección 3.2.2. El segundo exige una profundidad límite del 10% de la profundidad máxima de todo el archivo, quitando así cualquier registro de lecturas que están fuera de la región de interés, que son secuenciadas de manera indeseada. Estas regiones son llamadas regiones *off target*.

### 3.3.1. Detector de Cambios de Aminoácidos

El paquete de Bioconductor llamado “Biostrings” tiene muchas funciones para trabajar con secuencias de ADN, ARN y aminoácidos. Se usaron estas herramientas para detectar cambios de aminoácidos en variantes detectadas por los *variant callers*.

Las funciones relevantes para el proyecto fueron:

- **translate** - Toma una secuencia de ADN o ARN y la traduce a aminoácidos.
- **replaceLetterAt** - Reemplaza un componente de la secuencia en una posición especificada.
- **matchPattern** - Realiza una matriz de alineamiento entre secuencias.
- **mismatch** - Devuelve las posiciones que variaciones entre dos secuencias.

### 3.4. Evaluación de Variant Callers con Datos Artificiales

Obtener un gold standard de mutaciones de baja frecuencia resulta muy difícil, debido a que sólo se podría obtener haciendo un estudio confirmando la presencia fenotípica de una mutación no silenciosa. En el caso particular de éste trabajo, sería confirmar la resistencia a los medicamentos empíricamente en el tratamiento médico. Recopilar este tipo de dato tomaría años y una inversión considerable en investigación clínica. Es por eso que se decidió simular datos para obtener un gold standard de variantes conocidas insertadas deliberadamente.

Se listaron mutaciones asociadas a resistencia reportadas en la bibliografía [47]. Las mismas están a continuación en el cuadro 3.1 y fueron insertadas en el genoma de referencia de HCV 1b, y se generaron archivos fastq artificiales usando el Artificial-FastqGenerator. Si bien las mutaciones del cuadro son 27, dos de ellas, L31M y L31V de la proteína NS5A, corresponden a un cambio en la misma posición. La diferencia entre ambas es que una cambia a adenina y la segunda a guanina. Debido a que el rendimiento de los *variant callers* no cambia según el tipo de base, resulta redundante tomar en cuenta ambas mutaciones y se escogió utilizar solamente la mutación L31M. Se prepararon cuatro muestras, cada una conteniendo las 26 mutaciones con una misma frecuencia alélica en cada posición. Los AF fueron: 0.2, 0.05, 0.01 y 0.005. Los datos artificiales se generaron con errores aleatorios simulados de secuenciación, y se escogió una profundidad de 1000, para que la menor frecuencia escogida, de 0.005 sea muy cercana al error simulado (5x vs. 1x).

Una vez obtenidos los FASTQ simulados, se alinearon con BWA MEM y se obtuvieron los archivos BAM con Samtools. Estos archivos BAM fueron entonces procesados por cada uno de los 6 *variant callers*, cada uno produciendo su correspondiente archivo

Proteína	Mutación	Codón Original	Codón Mutado	Ref	Alt	Posición Proteína	Posición Genoma
NS3	V36I	GTC	ATC	G	A	106	3525
	T54S	ACT	TCT	A	T	160	3579
	Q80K	CAG	AAG	C	A	238	3657
	Q80R	CAG	CGG	A	G	239	3658
	S122T	AGC	ACC	G	C	365	3784
	D168E	GAC	GAG	C	G	504	3923
	M175L	ATG	CTG	A	C	523	3942
NS5A	L31M	TTG	ATG	T	A	91	6348
	L31V	TTG	GTG	T	G	91	6348
	L31I	TTG	ATT	G	T	93	6350
	P58S	CCA	TCA	C	T	172	6429
	E62D	GAG	GAT	G	T	186	6443
	A92T	GCG	ACG	G	A	274	6531
	Y93H	TAC	CAC	T	C	277	6534
NS5B	S82T	CCT	ACT	C	A	241	7842
	L159E	CTT	GAA	C	G	472	8073
	L159E	CTT	GAA	T	A	473	8074
	L159E	CTT	GAA	T	A	474	8075
	C316Y	TGC	TAC	G	A	944	8545
	M414I	ATG	ATC	G	C	1239	8840
	C445F	TGT	TTT	G	T	1331	8932
	Y448C	TAC	TGC	A	G	1340	8941
	C451S	TGT	TCT	G	C	1349	8950
	A553V	GCT	GTT	C	T	1655	9256
	S556G	AGC	GGC	A	G	1663	9264
	S556R	AGC	AGG	C	G	1665	9266
	I585V	ATC	GTC	A	G	1750	9351

**Cuadro 3.1.:** Listado completo de variantes asociadas a resistencia encontradas en el VHC tipo 1b, según Chen et al. 2016

VCF, listando las variantes encontradas. Teniendo en cuenta que son 6 resultados por cada muestra, y se generaron 4 muestras artificiales, se procesaron un total de 24 resultados de llamado de variantes para los datos artificiales.

Cada set de 6 archivos VCF fue procesado por separado, de acuerdo a los AF de cada uno. En el proceso se extrae, usando un programa en R, usando herramientas del paquete “Variant Annotation”, las posiciones de las variantes detectadas por cada *variant caller*. Estas posiciones son comparadas con el set de 26 mutaciones del gold standard, y a partir de esta comparación se obtienen los verdaderos positivos detectados para cada *variant caller*. Se obtiene la sensibilidad al dividir el número de

verdaderos positivos por el número de mutaciones del gold standard (que son siempre 26, ecuación 3.1). También se calcula el valor predictivo positivo (PPV), que resulta de dividir el número de verdaderos positivos por el número de posiciones detectadas por ese *variant caller* (ecuación 3.2).

$$sens = \frac{VerdaderosPositivos}{Nro.VariantesGoldStandard} \quad (3.1)$$

$$PPV = \frac{VerdaderosPositivos}{Nro.VariantesTotalesDetectadas} \quad (3.2)$$

Se calcula también el F1 score, que evalúa el rendimiento general de los *variant callers*, considerando tanto la sensibilidad como el PPV con el mismo peso.

$$F1 = \frac{2 * sens * PPV}{sens + PPV} \quad (3.3)$$

Los resultados de sensibilidad y PPV para cada frecuencia alélica se vuelcan en una tabla. Estas tablas se pueden encontrar en el apéndice A, sección 2.

Por último usando `ggplot2`, un paquete de herramientas de visualización en R, se realizan visualizaciones de los datos. Se grafican las posiciones detectadas por cada *variant caller* en un *stripchart*, otra herramienta de R para distribuir puntos a lo largo de una recta, que permite contemplar las posiciones en escala (en el capítulo 4, se aprecian en la Figura 4.1). También se hacen gráficos de dispersión de sensibilidad vs. PPV de los *variant callers* para los cuatro set de datos con distinto AF. Y por último se combinan los datos de las cuatro tablas para realizar gráficos de línea, en los que se analizan los cambios en Sensibilidad vs. AF, PPV vs. AF y sobre todo, F1 score vs. AF. Los gráficos y tablas resultantes serán expuestas en el capítulo de Resultados.

## 3.5. Evaluación de Variant Callers con Datos Reales

### 3.5.1. Gold Standard Basado en Variantes Asociadas de Resistencia

Para los datos reales no tenemos un gold standard, lo cual complica una evaluación tan equitativa como en las pruebas con datos artificiales. Si se sabe que existen posiciones que provocan resistencia a drogas, y si en estas muestras hay una variante presente en las posiciones asociadas a resistencia, es muy probable que estas variantes sean reales y sean de alto riesgo.

Por eso se tomaron las 26 variantes usadas en el tratamiento con datos artificiales, y se buscó cual coincide con las posiciones encontradas en el *naive variant caller*, que tiene siempre una sensibilidad absoluta por diseño. Si en las variantes detectadas por el *naive variant caller* se encuentra alguna en estas 26 posiciones, se plantea la asunción de que estas variantes son verdaderas y por lo tanto un gold standard impuesto. En otras palabras, se evalúa que tan buenos son los *variant callers* en detectar variantes en posiciones de interés. La consecuencia de esto es que el *naive variant caller* se coloca en una posición privilegiada de ser casi un estándar de referencia, lo cual no es necesariamente real pero si se puede observar los rendimientos del resto de los *variant callers* comparativamente.

Las muestras reales, como fue mencionado en la sección 3.1, corresponden a 16 muestras de la región correspondiente a la proteína NS3, y 16 a NS5B. Las regiones secuenciadas no cubren completamente la región codificante, es por eso que no todas las variantes asociadas a resistencia del cuadro 3.1 son candidatas.

Ambos sets de 16 muestras son procesadas por separado y luego sus resultados sumados. También se procesan con el paquete Variant Annotation en R. Ahora la evaluación de sensibilidad, valor predictivo positivo y F1 score resulta un poco mas compleja. La sensibilidad de un variant caller viene a ser, como se muestra en la ecuación 3.4 la suma de los verdaderos positivos (VP) encontrados en cada muestra por ese *variant caller*, dividido las variantes asociadas a resistencia encontradas por el *naive variant caller* en esa misma muestra (CP, sigla en ingles de *Condition Positive* que significa positivos de la condicion). Y el valor predictivo positivo, observando la ecuación 3.5, viene a ser la suma de verdaderos positivos detectados en cada muestra, dividido la suma del numero total de variantes detectadas por el *variant caller* en cada

muestra (PP, sigla en inglés de *predicted positive*, que significa positivos detectados). El cálculo del F1 score se mantiene igual, aplicado a la sensibilidad y PPV de estos casos.

$$sens = \frac{\sum_{i=1}^{16} VP_i}{\sum_{i=1}^{16} CP_i} \quad (3.4)$$

$$PPV = \frac{\sum_{i=1}^{16} VP_i}{\sum_{i=1}^{16} PP_i} \quad (3.5)$$

$$F1 = \frac{2 * sens * PPV}{sens + PPV} \quad (3.6)$$

En los datos reales cada variante detectada tiene su AF particular, es por eso que para analizar el rendimiento de detección con respecto a la reducción en AF ahora se tienen que filtrar las variantes detectadas a umbrales de AF determinados. También es necesario recalcular el CP y PP, en base a las variantes detectadas en el *naive variant caller* filtradas al umbral AF deseado para CP, y el total de variantes detectadas por el *caller* filtradas también al umbral de AF. Estos umbrales de AF coinciden con las AF insertadas artificialmente, que son 0.2, 0.05, 0.01 y 0.005. Además se observan los resultados con AF=1.0, es decir, viendo todas las variantes detectadas sin importar su frecuencia. Con los datos reales también se arma una tabla sumando los datos generales, que tiene como columnas, VP, PP, CP, la sensibilidad, el PPV y el F1 score, y en las filas, cada *variant caller*. Se repite esto para 5 umbrales de AF filtrados.

### 3.5.2. Gold Standard Basado en Variantes con Cambio de Sentido

Se realizó una segunda prueba con los mismos datos reales, un poco más especulativa y usando un gold standard nuevo, basado en todas las variantes que producen un cambio de sentido. Una variante con cambio de sentido significa que en la traducción de esta nueva secuencia se produce un cambio de aminoácido en el codón correspondiente. Entonces se busca que variantes detectadas en una de las muestras del *naive variant caller* provocan un cambio de aminoácidos y se define este conjunto como un nuevo gold standard. Este nuevo gold standard es considerablemente distinto

al anterior. En el anterior se buscaba en las variantes detectadas cual coincidía con tan solo 26 posiciones, y asumiendo que se encontrasen las 26 posiciones en los 16 archivos, resulta en un máximo de 416 posibles candidatos para el gold standard acumulado(CP). En este nuevo caso se evalúa cada una de las variantes detectadas por el *naive variant caller* y se verifica si producen un cambio de aminoácido. Si es que resulta en un cambio, entonces se agrega al gold standard. En este caso el gold standard es mucho mas amplio, con cientos de variantes por muestra y un gold standard acumulado en el orden de los miles.

Este cambio de aminoácidos es detectado con herramientas del paquete Biostrings en R, mencionado en la subsección 3.3.1. Al igual que en el anterior tratamiento con datos reales, esta técnica también coloca al *naive variant caller* en una posición privilegiada en la evaluación de rendimiento, ya que muchas de estas variantes, aunque tengan cambio de sentido, puede que no sean reales y al tomarse como reales por definición elevan los resultados de Sensibilidad, PPV y F1 del *naive variant caller*.

Debido a que este procesamiento es intenso y consiste de considerables cálculos, la capacidad para evaluar cada una de las 16 muestras supera la de un procesador convencional en la actualidad y requiere de tiempos que llegan a estar en el orden de días. Analizando las variantes coincidentes entre las 16 muestras, se evidenció una alta coincidencia entre las mismas. Por lo tanto se escogió la muestra con mayor número de variantes, se buscó las mutaciones con cambios de sentido en solamente ésta, y se tomó ese conjunto como el gold standard para todas las muestras.

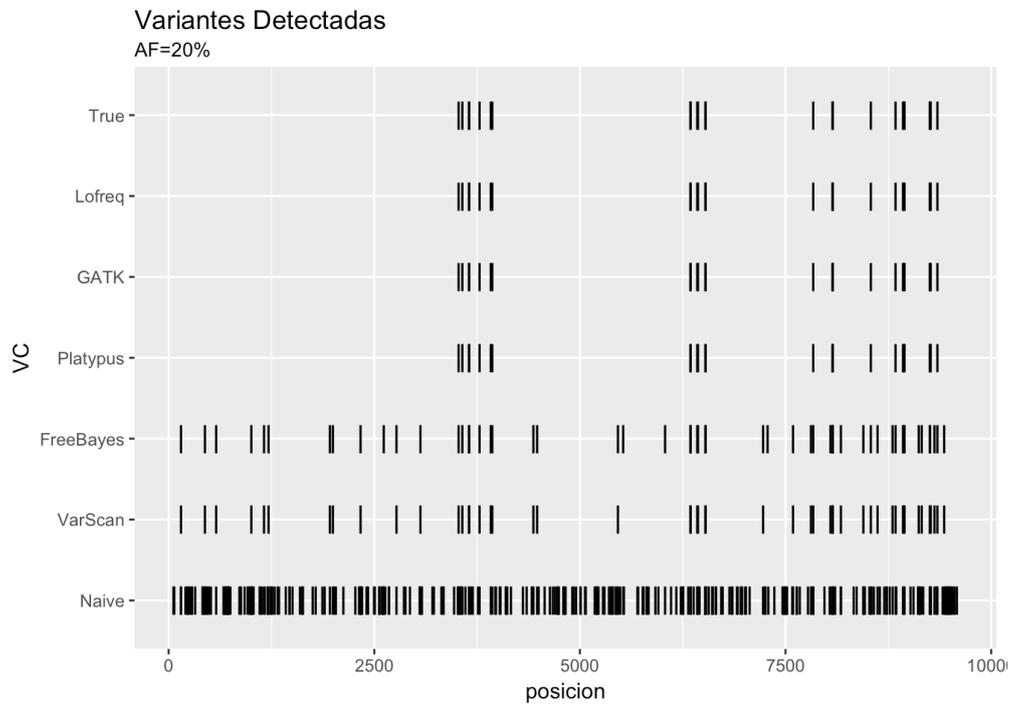
Aparte del cambio de gold standard, la evaluación se realiza de manera idéntica al anterior tratamiento con datos reales. Se procesaron las mismas 32 muestras correspondientes al genoma que codifican a las proteínas NS3 y NS5B. Los cálculos de Sensibilidad, PPV y F1 también son idénticos a los propuestos en las ecuaciones 3.4,3.5 y 3.6. El filtrado por umbrales de AF también se realiza de forma igual al anterior proceso.

## 4. Resultados y Discusión

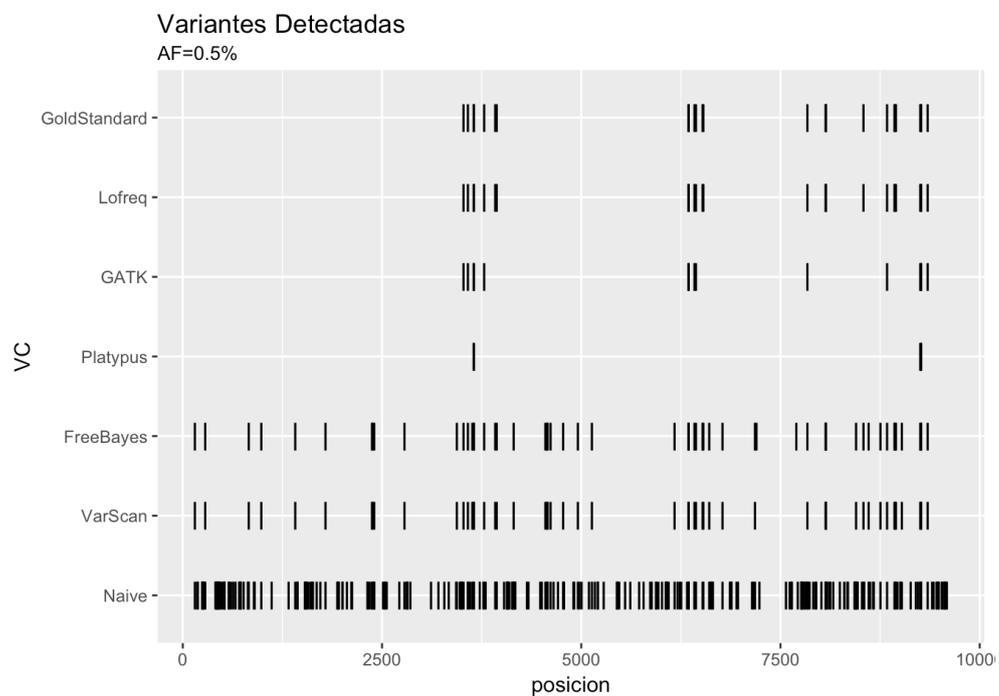
### 4.1. Resultados de Rendimiento de Detección con Datos Artificiales

Una vez obtenidos los parámetros de sensibilidad, valor predictivo positivo y F1 score para cada *variant caller*, se puede apreciar el comportamiento de estas herramientas ante los cambios que trae la diferencia de frecuencia alélica en cada variante. Los datos numéricos en tablas se encuentran en la sección 2 del apéndice. En la figura 4.1 se muestra una representación escalada de las posiciones de las variantes detectadas. Se muestran solo dos de los cuatro casos de AF, el máximo (de 20 %) y el mínimo (de 0,05 %), que tiene el mayor contraste para así poder apreciar los cambios la capacidad de detección en relación a las posiciones de las variantes.

Se aprecia de las gráficas de la figura 4.1 como algunos *variant callers* (GATK, Platypus) pasan a producir falsos negativos en la menor frecuencia, cuando en 20 % ninguno lo hacía. También se puede ver un aumento en falsos positivos por parte de FreeBayes y VarScan ante la reducción del AF. Estas son señales de lo que prioriza cada herramienta a la hora de decidir tomar una variante o no y considerarla una detección positiva.

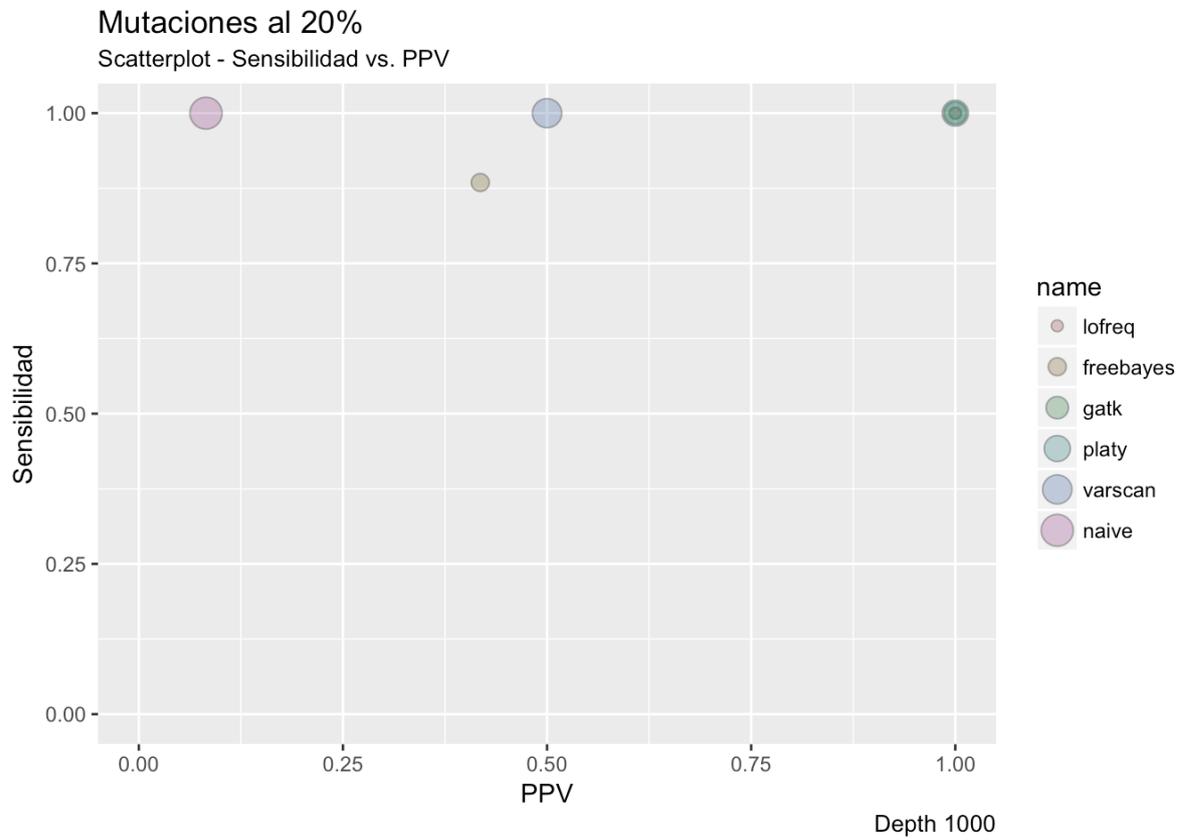


(a) AF=20 %

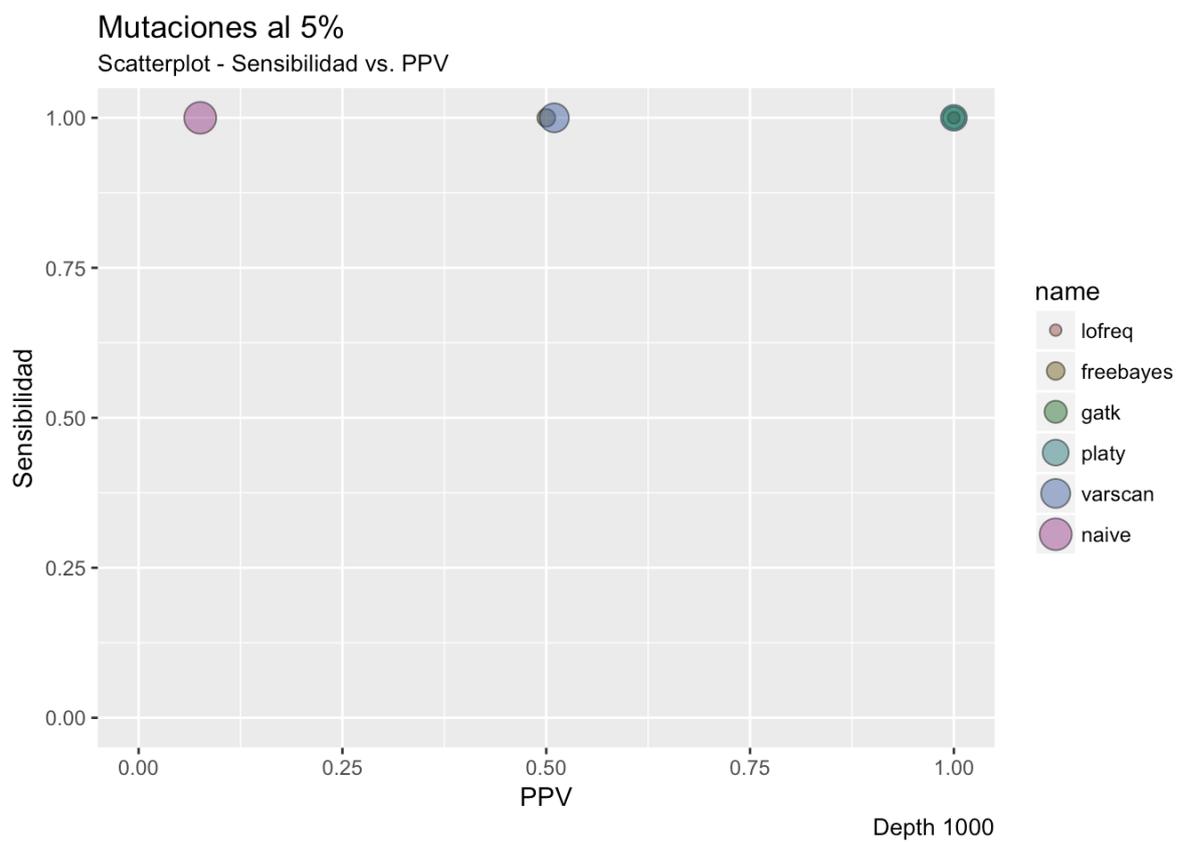


(b) AF=0,05 %

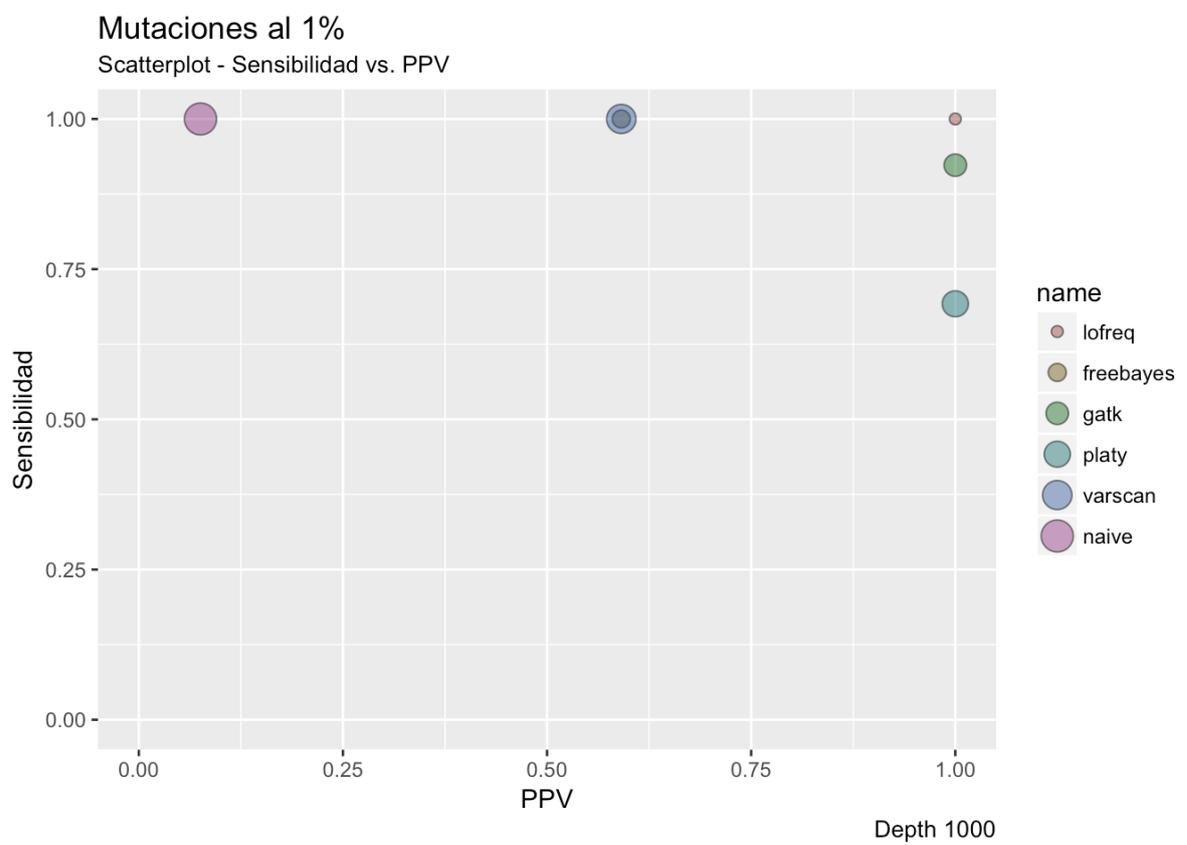
**Figura 4.1.:** Visualización de posiciones de variantes detectadas por cada variant caller. Se visualizan los casos para el mayor y el menor AF (20 % y 0,05 %, respectivamente)



**Figura 4.2.:** Diagrama de dispersión de sensibilidad vs. PPV. de AF=20%; GATK, Platypus y Lofreq superpuestos en la coordenada (1,1)



**Figura 4.3.:** Diagrama de dispersión de sensibilidad vs. PPV. de AF=5 %; GATK, Platypus y Lofreq superpuestos en la coordenada (1,1)



**Figura 4.4.:** Diagrama de dispersión de sensibilidad vs. PPV. de AF=1 %; FreeBayes y VarScan superpuestos

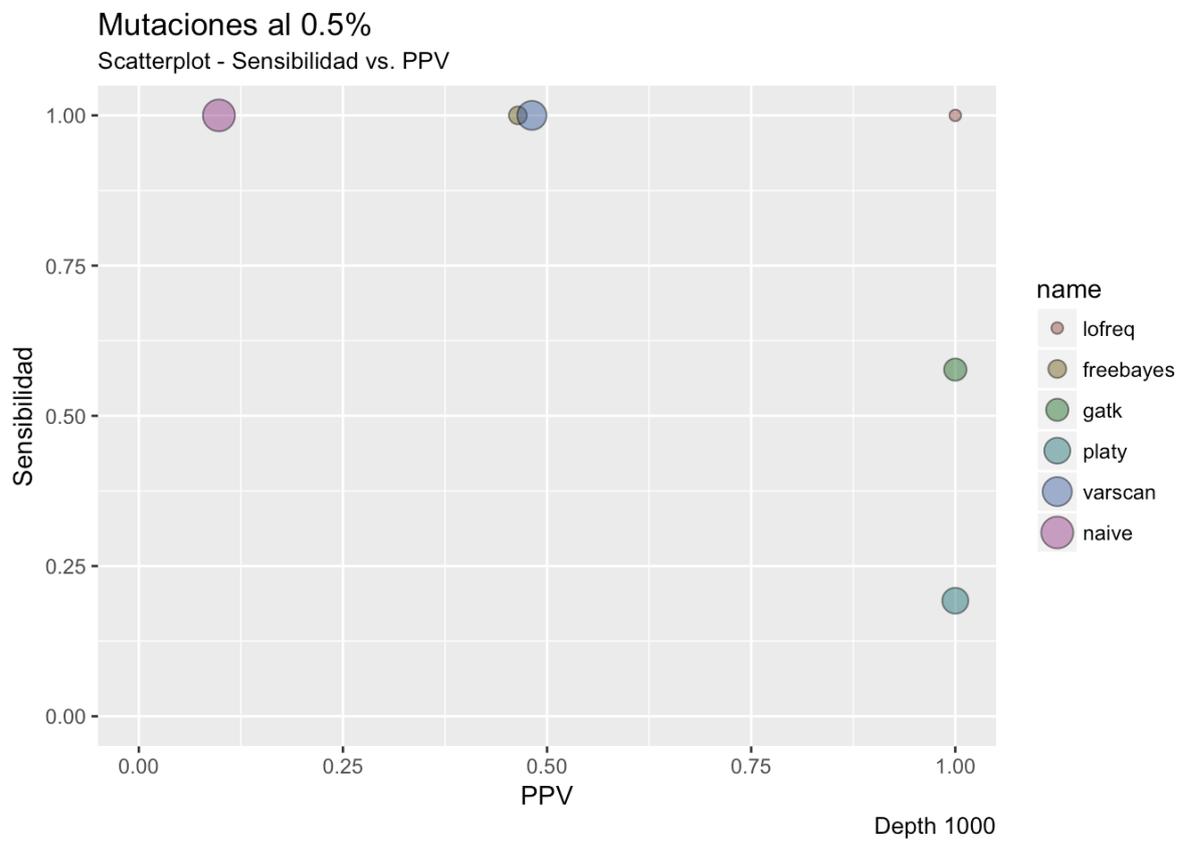


Figura 4.5.: Diagrama de dispersión de sensibilidad vs. PPV. de AF=0,05 %

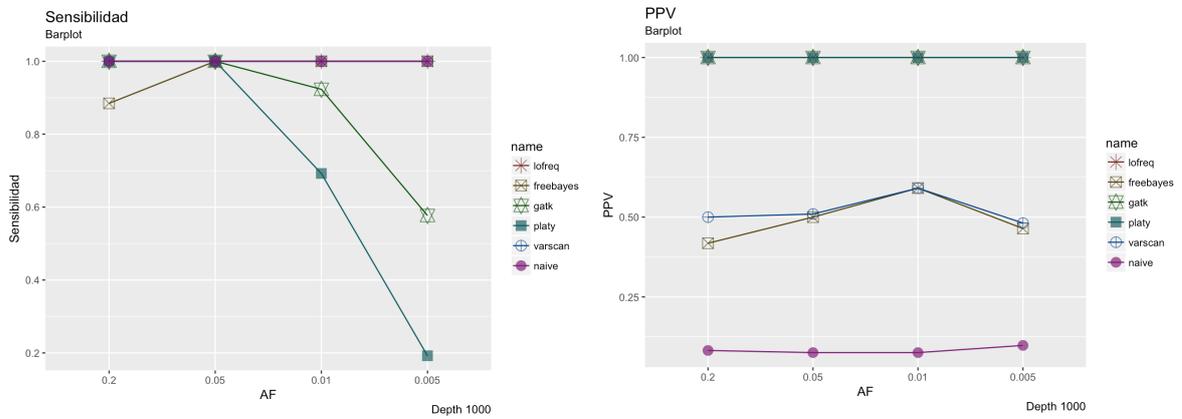
En las figuras 4.2,4.3,4.4 y 4.5 está graficada la desviación de los *variant callers* en su relación sensibilidad-PPV, una clara señal de que los datos son artificiales es que casi siempre los puntos se encuentran en los bordes de esta relación, es decir casi siempre con sensibilidad y/o PPV de valor 1. Ya desde una AF de 20 %, se evidencia que VarScan y naive tienen un PPV bajo, es decir que detectan muchos falsos positivos, el naive siendo más drástico en esto, como es de esperar. Aún así todos tienen una sensibilidad perfecta, excepto FreeBayes que falla en detectar 3 variantes. En el gráfico correspondiente a 5 %, las características se mantienen, pero FreeBayes ahora sensa todas las mutaciones insertadas, lo cual podría ser una señal de una falla circunstancial del algoritmo en la detección de variantes al 20 %.

Con respecto a los valores de 1 %, dos *variant callers*, GATK y Platypus, pierden sensibilidad, denotando su poca capacidad de rendir en muy bajas frecuencias. En el extremo de 0,05 %, la relación de condiciones en 1 % se mantienen, pero se profundizan la cantidad de falsos negativos para GATK y Platypus, y también la cantidad de falsos positivos de FreeBayes y Varscan.

En el caso de AF = 0,05 %, los reads por posición varían entre tan solo 4x-6x, cuando los errores simulados tienen una profundidad de 1x, claramente las variantes están muy cerca del ruido y resulta desafiante para un *variant caller* detectar todas estas sin atrapar algún falso positivo entre los errores simulados. Aún así en esta prueba, que se puede apreciar mejor en la figura 4.5 Lofreq lo logra, teniendo una sensibilidad y PPV perfectas, que también implica un F1 score de 1 en el extremo mínimo de fracciones alélicas simuladas.

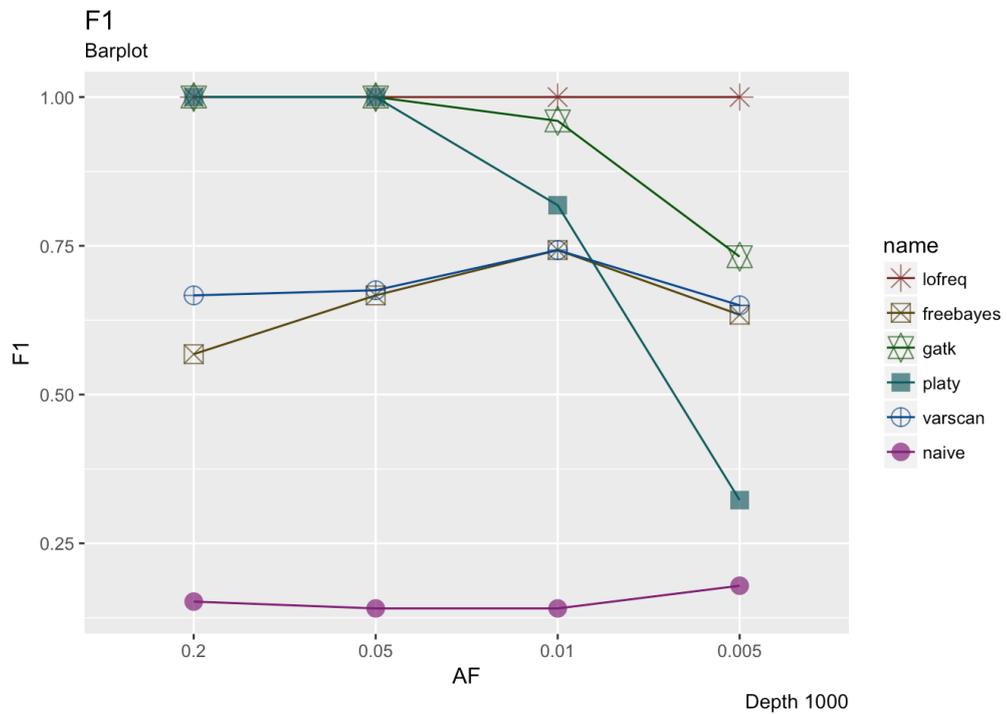
Los gráficos de la figura 4.6, muestran el cambio de cada característica evaluada con la reducción de AF. Aquí se puede apreciar las dos categorías de *variant callers* en las que caen estas herramientas: las que sacrifican falsos negativos a cambio de no tener falsos positivos (GATK y Platypus apreciado en la figura 4.6a), y las que sacrifican falsos positivos a cambio de no tener falsos negativos (FreeBayes, VarScan y Naive, apreciado en la figura 4.6b).

Por último, la evolución del F1 score con la reducción de AF (figura 4.6c) resulta un amalgamador coherente de todas las características previamente mencionadas, y también pone en evidencia la clara robustez de Lofreq en medio de todos los cambios.



(a) Sensibilidad

(b) PPV



(c) F1 score

Figura 4.6.: Sensibilidad, PPV y F1 score vs. AF

## 4.2. Resultados de Rendimiento de Detección con Datos Reales

### 4.2.1. Rendimiento con Gold Standard basado en Variantes Asociadas a Resistencia

Como en el análisis de datos reales el gold standard impuesto es un número muy reducido de variantes. En este análisis basado en variantes asociadas a resistencia, el gold standard acumulado consiste de tan solo entre 53 a 66 variantes para las 16 muestras, es decir entre 3 y 4 variantes encontradas en promedio por cada muestra, cuando los *variant callers* detectaron entre 373 (GATK) y 5325 (naive) en caso mínimo (AF= 0,05 %). La consecuencia es que los índices de sensibilidad y valor predictivo positivo resultan muy bajos, ya que los denominadores siempre resultan valores elevados con respecto a los verdaderos positivos de un gold standard tan reducido.

En la figura 4.7 se muestran los gráficos de sensibilidad vs. PPV para 5 AF's. Las 4 frecuencias usadas previamente en los datos y adicionalmente la de AF de 100 %, que muestra todas las variantes detectadas. Los gráficos están enfocados en sensibilidad entre 0-0,25 y PPV de 0-0,01 para poder apreciar mejor la dinámica del cambio de posiciones. Para hacer esto los resultados del Naive Variant Caller y VarScan quedan fuera del cuadro, pero más adelante se verá que ambos tienen una detección casi idéntica y al estar el Naive con una ventaja por ser referencia del gold standard no es de interés mostrarlos.

Lo más notorio en este caso es que Platypus está simplemente en 0, señal de que no detectó ninguna variante de interés y queda fuera del análisis. También se ve que para todos los umbrales de AF, Lofreq tiene superior sensibilidad pero menor o casi igual PPV que GATK y FreeBayes. Se observa una tendencia común, la sensibilidad de cada uno descende entre 100 % hasta 5 % y luego se queda fijo, esto puede deberse que que directamente casi todas las variantes debajo de 5 % están también debajo de 0,5 %.

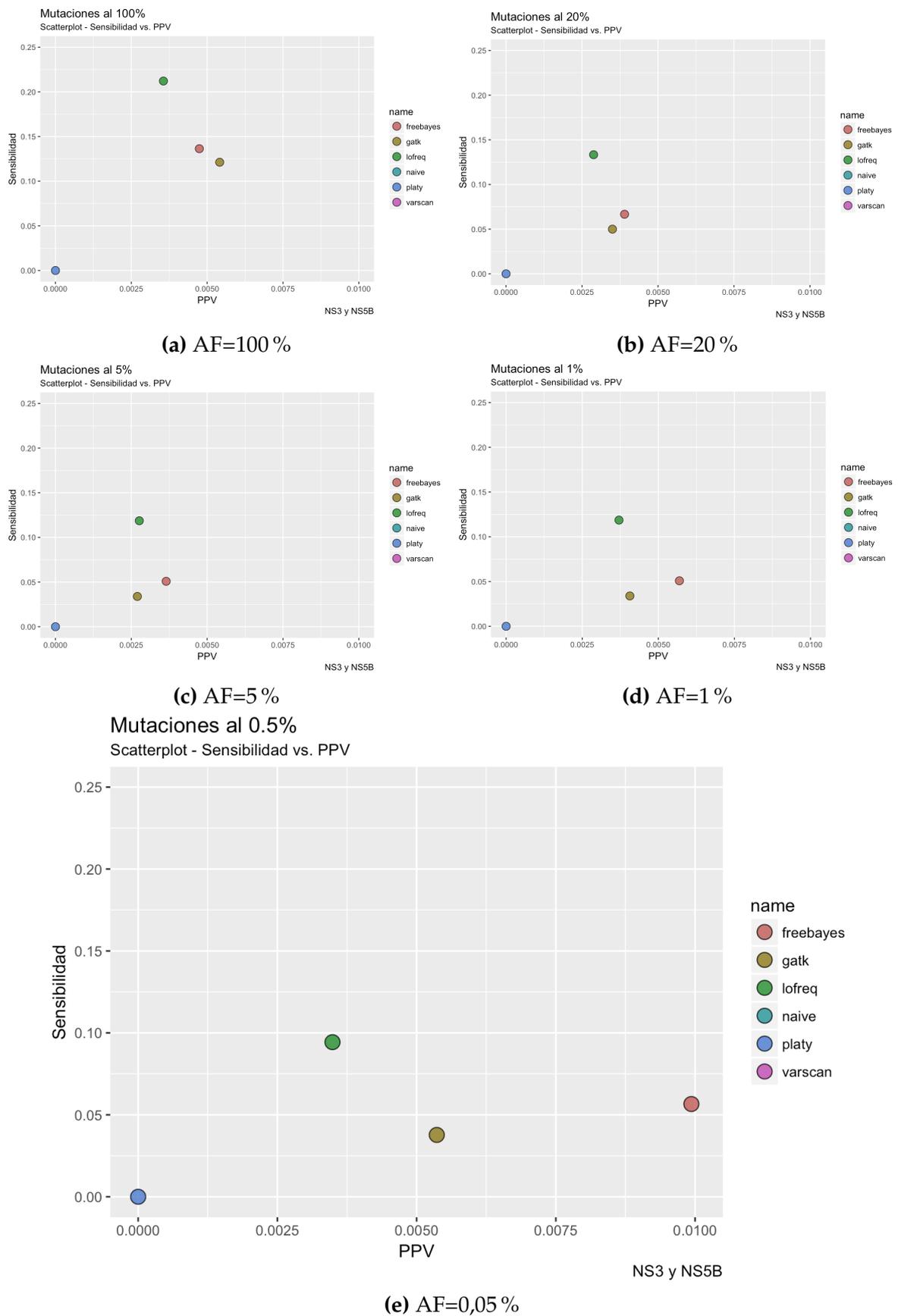
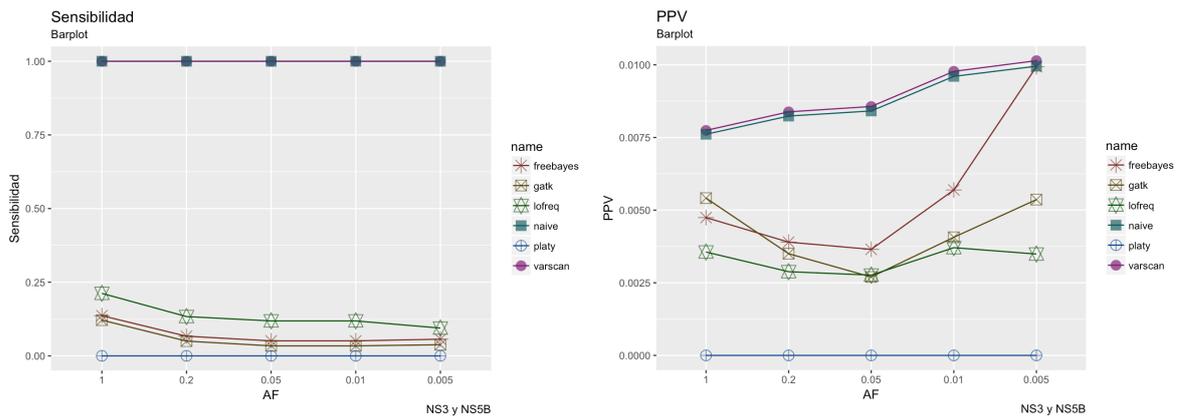


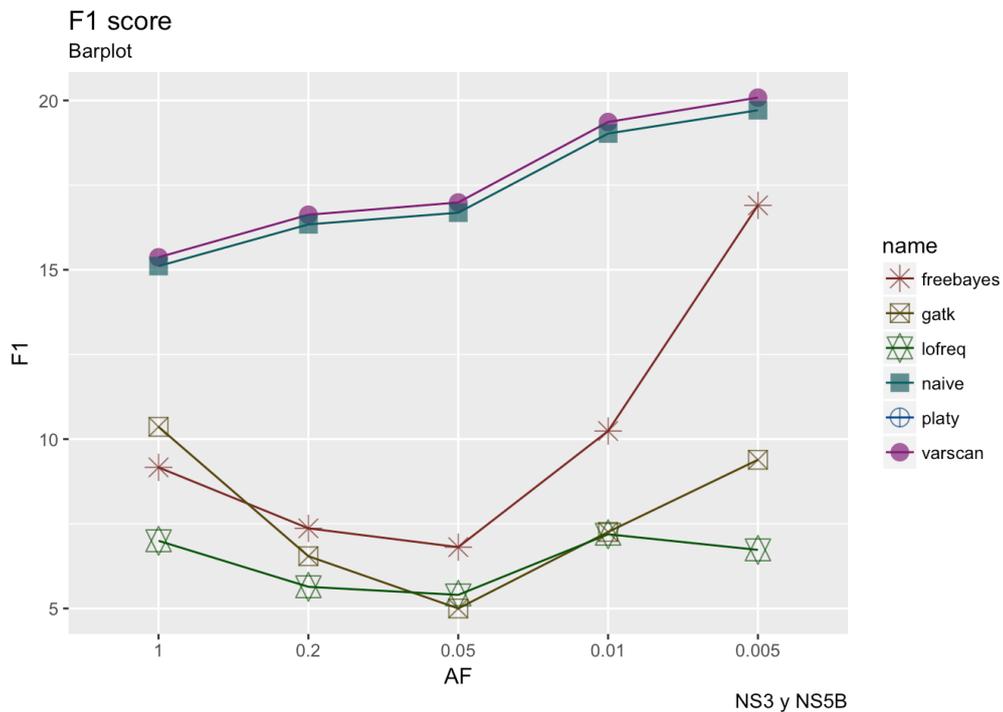
Figura 4.7.: Gráficos de dispersión de Sensibilidad vs. PPV.

En cuanto a PPV, GATK comienza siendo superior en AF=100 % pero en frecuencias menores a 20 %, FreeBayes toma la delantera y a medida que se desciende en AF, FreeBayes asciende en PPV hasta ser destacadamente el más alto. En la figura 4.7e que corresponde al gráfico con umbral de AF=0,5 %, se nota que Lofreq hasta el final sigue teniendo sensibilidad más alta y FreeBayes el PPV más alto.



(a) Sensibilidad

(b) PPV



(c) F1 score

Figura 4.8.: Sensibilidad, PPV y F1 score( $\times 10^3$ ) vs. AF

Observando la figura 4.8, se aprecia como evolucionan la Sensibilidad, PPV y F1 con filtraciones de AF cada vez menores. En el gráfico de F1 score, se multiplicó cada valor por mil, para facilitar la diferenciación de escalas nublada por largos números decimales. Viendo la sensibilidad se ven todos los *variant callers* estratificados, cada uno se queda en su lugar. Y también se aprecia como tanto el Naive y VarScan tienen sensibilidad de 1 siempre. Sin embargo, exceptuando estos dos, el de mayor sensibilidad es Lofreq. La situación de PPV es más compleja, pero se ve de nuevo el comportamiento similar de Naive con VarScan, nuevamente razón por la cual lo excluimos de la comparación por el resto al estar sesgados por estar referenciados al gold standard impuesto. Tomando lo anteriormente mencionado en cuenta, FreeBayes es claramente el caller con más alto PPV. Por último, viendo la evolución del F1 score, es muy similar al de PPV, solo eleva a Lofreq un poco con respecto a GATK y ambos casi se solapan entre 20% y 1%. De todos modos, es evidente que el F1 más alto es de GATK por sobre Lofreq en  $AF < 0,05\%$ .

#### 4.2.2. Rendimiento con Gold Standard basado en Variantes con Cambio de Sentido

En este caso los valores de sensibilidad, PPV y F1 score suben con respecto a la anterior prueba debido a que el gold standard es mucho más elevado. Los verdaderos positivos detectados por cada *variant caller* ascienden a los cientos (en la anterior prueba no superaban los 10) y eso impacta en el numerador del cálculo de tanto la sensibilidad como del valor predictivo positivo. Esta prueba viene a ser casi un control de la detección en los datos reales.

Los gráficos de dispersión de sensibilidad vs. PPV (figura 4.9) muestran a los *variant callers* en situaciones similares a la anterior prueba con datos reales. La diferencia más notoria es que ahora Platypus si está activo y, aunque tiene una sensibilidad muy baja, tiene un PPV que está en un rango parecido al de los demás *callers*. El cambio más notorio en las dispersiones con la reducción de filtrado de AF es que el PPV de todos los *callers* va aumentando.

La sensibilidad no parece cambiar mucho para todos los *callers*. Y una vez más el comportamiento de VarScan y Naive resultan casi idénticos. Otra diferencia con el anterior análisis es que ahora el Naive Variant Caller no tiene una sensibilidad de 1,

eso es porque el gold standard en esta ocasión se tomó de las variantes que provocan cambios de aminoácidos de una sola muestra de las 16.

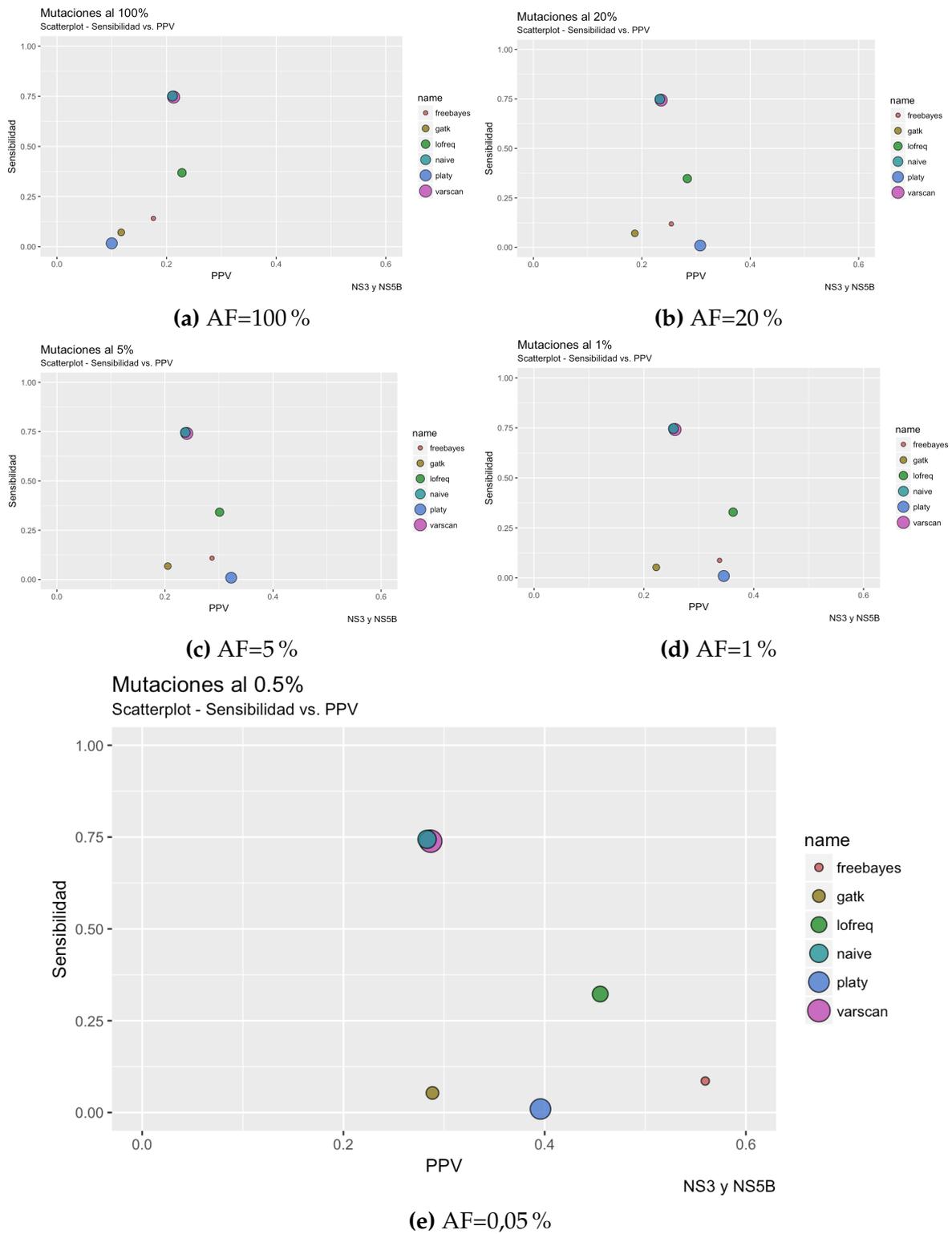
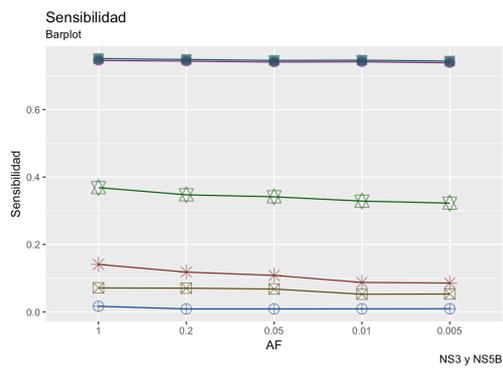


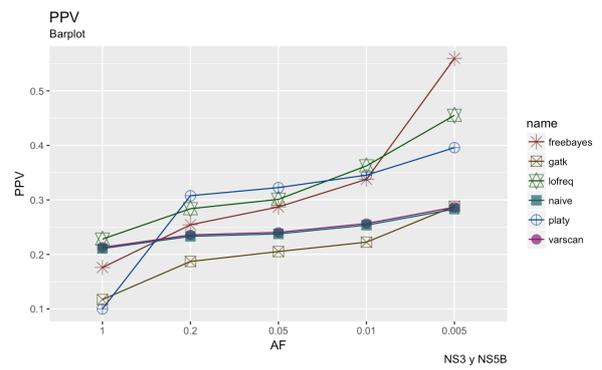
Figura 4.9.: Gráficos de dispersión de Sensibilidad vs. PPV.

Si se aprecia la figura 4.10, el cambio en la sensibilidad es parecido al visto en el anterior análisis, niveles estratificados para cada caller y en el mismo orden que el anterior análisis también. La evolución en PPV cambia, ya que en este caso no sólo todos suben constantemente, como se mencionó al comentar los gráficos de dispersión, si no que todos se comportan de forma muy parecida, como la anterior vez, FreeBayes resulta con mayor PPV a menor frecuencia, esta vez por encima de Naive y VarScan, y también lo hace Lofreq y Platypus.

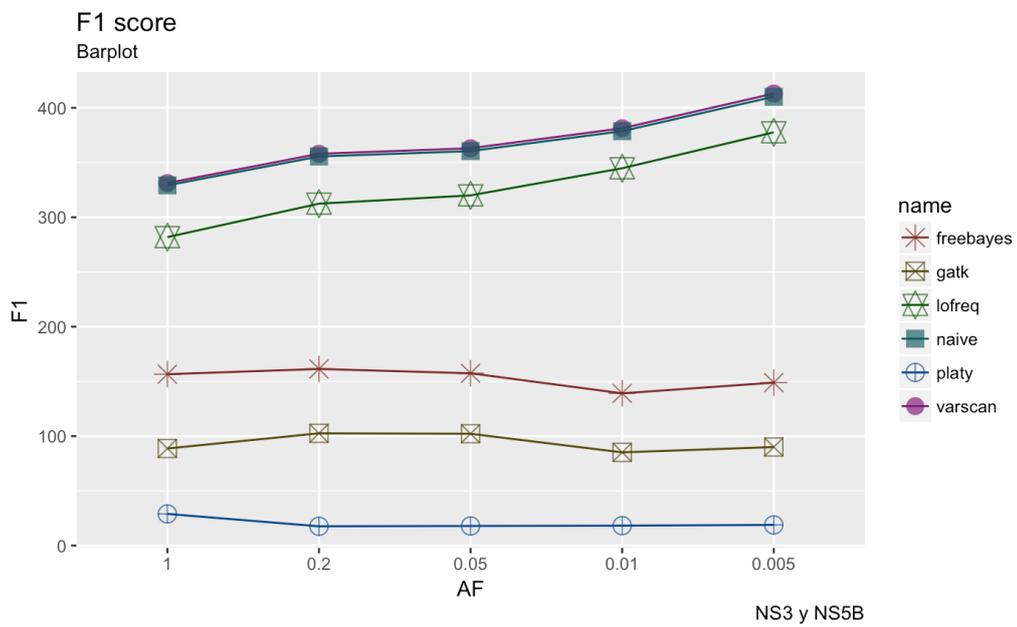
Y finalmente observando el gráfico del F1 score se ve que en efecto el comportamiento de todos los *variant callers* en esta prueba es muy similar, y el nivel esta casi completamente determinado por sus sensibilidades. En esta caso los valores de F1 también fueron multiplicados por mil para facilitar la visualización de los números. La explicación al comportamiento tan similar es que el gold standard en este caso es muy grande y casi aleatorio, de manera que se espera que no haya ningún criterio de detección especial en los algoritmos de cada *variant caller* que destaque y se pueda diferenciar. Aún así vale notar que, dejando a un lado el Naive Variant Caller y VarScan, Lofreq tiene la mayor sensibilidad y simultáneamente un mayor PPV para todas las frecuencias excepto 0,05% y eso indica que si capta muchas de las variantes, pero su sensibilidad menor pero proporcional al Naive dice que está rechazando muchas variantes detectadas por este, debido a que no cumplen con los mínimos de calidad.



(a) Sensibilidad



(b) PPV



(c) F1 score

Figura 4.10.: Sensibilidad, PPV y F1 score( $\times 10^{-3}$ ) vs. AF

## 5. Conclusiones

Se concluye, a partir del análisis de los datos artificiales, con variantes insertadas a distintas frecuencias alélicas, de que Lofreq es el *variant caller* que fue más apto para detectar las variantes de forma precisa. Seguido de este, GATK es el segundo más apto, aunque con sensibilidad comprometida en muy bajas frecuencias.

Apartir de lo visto en el análisis de datos reales, con gold standard basado en variantes asociadas a resistencia se concluye que FreeBayes es el *variant caller* con mejor rendimiento, por detectar un menor número de falsos positivos, lo cual eleva su valor predictivo positivo. En segundo lugar de esta evaluación, se concluye que Lofreq es el indicado, ya que aunque no supera a GATK en niveles de F1 score para la mayoría de las frecuencias, tiene una mayor sensibilidad, incluso que FreeBayes, lo cual tiene más peso en este análisis que el valor predictivo positivo.

En el análisis de datos reales basado en las variantes asociadas a resistencia se observa que para el caso de Lofreq, el algoritmo con mayor sensibilidad, las variantes encontradas en total fueron 14. Pero las variantes con frecuencias de 20 % o más, que son las únicas que se podrían detectar por el método Sanger, son tan solo 7. La mitad de las variantes de riesgo en este conjunto de muestras se encuentran por debajo del 5 % y son apreciables debido a la secuenciación NGS.

Del análisis de datos reales con gold standard basado en variantes con cambio de sentido se concluye que el *variant caller* más apto es Lofreq nuevamente, que uniformemente tuvo un rendimiento más alto en F1 score, priorizando una sensibilidad mayor en toda frecuencia, lo cual es más significativo. En segundo lugar, estaría FreeBayes que muestra las mismas características mencionadas para Lofreq en este caso, pero con rendimientos levemente inferiores, particularmente en sensibilidad.

Del conjunto de los tres análisis, se concluye que Lofreq es la herramienta más apta para poder sensar todas las variantes reales de baja frecuencia, con extremadamente bajo riesgo de detectar falsos negativos, y simultáneamente con muy bajos pero presentes niveles de falsos positivos.

# Apéndice A.

## Información Adicional

### A.1. Lista de Datos Seleccionados de Base de Datos de NCBI

A continuación los códigos para descargar los archivos FASTQ del BioProject PRJNA433160 con SRA. La lista completa de las muestras de este estudio se pueden acceder con el link: <https://www.ncbi.nlm.nih.gov/Traces/study/?acc=SRP133915>

**Muestras de NS3:** SRR6805024, SRR6805022, SRR6805020, SRR6805017, SRR6805016, SRR6805014, SRR6805012, SRR6805010, SRR6805008, SRR6805006, SRR6805005, SRR6805003, SRR6805001, SRR6804999, SRR6804997, SRR6804994

**Muestras de NS5B:** SRR6805025, SRR6805023, SRR6805021, SRR6805019, SRR6805009, SRR6805007, SRR6805004, SRR6805002, SRR6805000, SRR6804998, SRR6804996, SRR6804995, SRR6804993, SRR6804991, SRR6804988, SRR6804987

### A.2. Tablas de Resultados de Evaluación de Datos Artificiales

A continuación se muestran las tablas con los valores de rendimiento ( sensibilidad, valor predictivo positivo y F1 score) de los variant caller evaluados para las mutaciones de cada fracción alélica.

---

<b>Variantes con AF=20 %</b>			
<b>Variant Caller</b>	<b>sens</b>	<b>PPV</b>	<b>F1</b>
Lofreq	1,00	1,00	1,00
FreeBayes	0,88	0,42	0,57
GATK	1,00	1,00	1,00
Platypus	1,00	1,00	1,00
VarScan	1,00	0,50	0,67
Naive	1,00	0,08	0,15

---



---

<b>Variantes con AF=5 %</b>			
<b>Variant Caller</b>	<b>sens</b>	<b>PPV</b>	<b>F1</b>
Lofreq	1,00	1,000	1,00
FreeBayes	1,00	0,500	0,67
GATK	1,00	1,000	1,00
Platypus	1,00	1,000	1,00
VarScan	1,00	0,509	0,67
Naive	1,00	0,076	0,14

---



---

<b>Variantes con AF=1 %</b>			
<b>Variant Caller</b>	<b>sens</b>	<b>PPV</b>	<b>F1</b>
Lofreq	1,00	1,000	1,00
FreeBayes	1,00	0,591	0,74
GATK	0,92	1,000	0,96
Platypus	0,69	1,000	0,82
VarScan	1,00	0,591	0,74
Naive	1,00	0,076	0,14

---

Variantes con AF=0.5 %			
Variant Caller	sens	PPV	F1
Lofreq	1,00	1,000	1,00
FreeBayes	1,00	0,464	0,63
GATK	0,58	1,000	0,73
Platypus	0,19	1,000	0,32
VarScan	1,00	0,481	0,65
Naive	1,00	0,098	0,18

### A.3. Tablas de Resultados de Evaluación de Datos Reales

#### A.3.1. Gold Standard Basado en Variantes Asociadas a Resistencia

Variantes con AF<100 %						
Variant Caller	VP	PP	CP	sens	PPV	F1
Lofreq	14	3935	66	0,21	0,0036	6,99
FreeBayes	9	1898	66	0,14	0,0047	9,16
GATK	8	1478	66	0,12	0,0054	10,36
Platypus	0	407	66	0,00	0,0000	0,00
VarScan	66	8523	66	1,00	0,0077	15,37
Naive	66	8671	66	1,00	0,0076	15,11

---

Variantes con AF<20 %						
Variant Caller	VP	PP	CP	sens	PPV	F1
Lofreq	8	2777	61	0,13	0,0029	5,64
FreeBayes	4	1026	61	0,07	0,0039	7,36
GATK	3	857	61	0,05	0,0035	6,54
Platypus	0	66	61	0,00	0,0000	0,00
VarScan	61	7158	61	1,00	0,0085	16,90
Naive	61	7284	61	1,00	0,0084	16,61

---



---

Variantes con AF<5 %						
Variant Caller	VP	PP	CP	sens	PPV	F1
Lofreq	7	2533	59	0,12	0,0028	5,40
FreeBayes	3	822	59	0,05	0,0036	6,81
GATK	2	741	59	0,03	0,0027	5,00
Platypus	0	63	59	0,00	0,0000	0,00
VarScan	59	6887	59	1,00	0,0086	16,99
Naive	59	7013	59	1,00	0,0084	16,69

---



---

Variantes con AF<1 %						
Variant Caller	VP	PP	CP	sens	PPV	F1
Lofreq	7	1888	59	0,12	0,0037	7,19
FreeBayes	3	527	59	0,05	0,0057	10,24
GATK	2	492	59	0,03	0,0041	7,26
Platypus	0	56	59	0,00	0,0000	0,00
VarScan	59	6035	59	1,00	0,0098	19,36
Naive	59	6144	59	1,00	0,0096	19,02

---

Variantes con AF<0,5 %						
Variant Caller	VP	PP	CP	sens	PPV	F1
Lofreq	5	1433	53	0,09	0,0035	6,73
FreeBayes	3	302	53	0,06	0,0099	16,90
GATK	2	373	53	0,04	0,0054	9,39
Platypus	0	49	53	0,00	0,0000	0,00
VarScan	53	5225	53	1,00	0,0101	20,08
Naive	53	5325	53	1,00	0,0100	19,71

### A.3.2. Gold Standard Basado en Variantes con Cambio de Sentido

Variantes con AF<100 %						
Variant Caller	VP	PP	CP	sens	PPV	F1
Lofreq	873	3827	2368	0,369	0,228	0,282
FreeBayes	334	1898	2368	0,141	0,176	0,157
GATK	169	1442	2368	0,071	0,117	0,089
Platypus	40,00	400,00	2.368,00	0,017	0,100	0,029
VarScan	1.766,00	8.294,00	2.368,00	0,746	0,213	0,331
Naive	1.779,00	8.441,00	2.368,00	0,751	0,211	0,329

---

Variantes con AF<20 %						
Variant Caller	VP	PP	CP	sens	PPV	F1
Lofreq	767	2702	2208	0,347	0,284	0,312
FreeBayes	261	1026	2208	0,118	0,254	0,161
GATK	156	834	2208	0,071	0,187	0,103
Platypus	20	65	2208	0,009	0,308	0,018
VarScan	1642	6965	2208	0,744	0,236	0,358
Naive	1653	7090	2208	0,749	0,233	0,356

---



---

Variantes con AF<5 %						
Variant Caller	VP	PP	CP	sens	PPV	F1
Lofreq	743	2468	2176	0,341	0,301	0,320
FreeBayes	236	822	2176	0,108	0,287	0,157
GATK	148	721	2176	0,068	0,205	0,102
Platypus	20	62	2176	0,009	0,323	0,018
VarScan	1612	6704	2176	0,741	0,240	0,363
Naive	1623	6829	2176	0,746	0,238	0,360

---



---

Variantes con AF<1 %						
Variant Caller	VP	PP	CP	sens	PPV	F1
Lofreq	668	1843	2032	0,329	0,362	0,345
FreeBayes	178	527	2032	0,088	0,338	0,139
GATK	107	481	2032	0,053	0,222	0,085
Platypus	19	55	2032	0,009	0,345	0,018
VarScan	1507	5872	2032	0,742	0,257	0,381
Naive	1517	5980	2032	0,747	0,254	0,379

---

---

Variantes con AF<0,5 %						
Variant Caller	VP	PP	CP	sens	PPV	F1
Lofreq	635	1395	1968	0,323	0,455	0,378
FreeBayes	169	302	1968	0,086	0,560	0,149
GATK	105	364	1968	0,053	0,288	0,090
Platypus	19	48	1968	0,010	0,396	0,019
VarScan	1454	5072	1968	0,739	0,287	0,413
Naive	1464	5171	1968	0,744	0,283	0,410

---

# Apéndice B.

## Programas

### B.1. Procesamiento en R del Naive Variant Caller

```
1 library("data.table")
2 library(Biostrings)
3
4
5 rm(list=ls())
6 setwd("~/Bioinf/realseq/anaive")
7
8 #Paso los txt a csv
9
10 FILES <- list.files(pattern = "sample1.txt")
11
12 #Comienzo a trabajar cada AF dataframe
13
14 callvariants<-function(filename){
15
16 FILE=read.table(file=filename,header=T)
17
18 allAs <- FILE$A+FILE$a
19 allTs <- FILE$T+FILE$t
20 allCs <- FILE$C+FILE$c
21 allGs <- FILE$G+FILE$g
22
```

```
23 DP <- allAs+allTs+allCs+allGs
24
25 pos <- FILE$loc
26 ref <- FILE$ref
27
28 Variants <- data.frame(pos, ref, allAs, allTs, allCs, allGs)
29
30 #Valores totales para cada nucleotido
31 AFsA <- allAs/DP
32 AFsT <- allTs/DP
33 AFsC <- allCs/DP
34 AFsG <- allGs/DP
35
36 #Armo tabla de AFs
37 afdF <- data.frame(pos, ref, AFsA, AFsT, AFsC, AFsG, DP)
38
39 # Los que tienen valor 1 los paso a 0
40 afdF$AFsA[afdF$AFsA==1.0]<- 0
41 afdF$AFsT[afdF$AFsT==1.0]<- 0
42 afdF$AFsC[afdF$AFsC==1.0]<- 0
43 afdF$AFsG[afdF$AFsG==1.0]<- 0 #despues anadir condicion de
    verificar referencia!
44
45
46 #Paso a 0 las frecuencias que son iguales a la referencia
47 for(i in 1:length(afdF$pos)){
48
49     if(afdF$ref[i]=="A"){
50         afdF$AFsA[i]<-0
51     }
52     if(afdF$ref[i]=="T"){
53         afdF$AFsT[i]<-0
54     }
55     if(afdF$ref[i]=="C"){
56         afdF$AFsC[i]<-0
57     }
}
```

```
58   if (afdf$ref[i]=="G"){
59     afdF$AFsG[i]<-0
60   }
61
62 }
63 #Obtengo la suma de cada frecuencia , si suma cero es pq habia
    un 1
64 AFsum<-afdf$AFsA+afdf$AFsT+afdf$AFsC+afdf$AFsG
65
66 #Paso a 0 los NA
67 AFsum[is.na(AFsum)] <- 0
68
69 #Inserto cmo columna a AFDF
70 afdF <- cbind(afdf ,AFsum)
71
72 #Elimino todas las que suman 0, marcadas como no variantes.
73 afdF<-afdf[!(afdf$AFsum==0) ,]
74
75 #Elimino todas las filas con DP<4000, marcadas como no
    variantes.
76 afdF<-afdf[!(afdf$DP<(max(DP)*0.05)) ,]
77
78 #Elimino la columna AFsum
79 afdF$AFsum<- NULL
80
81 #Paso a 0 las frecuencias menores a 0.001
82 for(i in 1:length(afdf$pos)){
83
84   if (afdf$AFsA[i]<0.001){
85     afdF$AFsA[i]<-0
86   }
87   if (afdf$AFsT[i]<0.001){
88     afdF$AFsT[i]<-0
89   }
90   if (afdf$AFsC[i]<0.001){
91     afdF$AFsC[i]<-0
```

```
92   }
93   if ( afdF$AFsG[ i ] < 0.001 ) {
94     afdF$AFsG[ i ] <- 0
95   }
96
97 }
98
99 #Obtengo la suma de cada frecuencia , si suma cero es pq habia
   un 1
100 AFsum <- afdF$AFsA + afdF$AFsT + afdF$AFsC + afdF$AFsG
101 #Paso a 0 los NA
102 AFsum[ is.na(AFsum) ] <- 0
103
104 #Inserto como columna a AFDF
105 afdF <- cbind(afdF , AFsum)
106
107 #Elimino todas las que suman 0, marcadas como no variantes.
108 afdF <- afdF[ !(afdF$AFsum == 0) , ]
109
110 #Elimino la columna AFsum
111 afdF$AFsum <- NULL
112 rm(FILE)
113 rm(Variants)
114 #Exportar el dataframe "afdF" a ".txt"
115 write.table(afdF , paste0(filename , ".naivecalls.txt") , sep = "\t"
   )
116 }
117 for ( i in 1:length(FILEs) ) {
118   callVariants(FILEs[ i ])
119 }
```

## B.2. Evaluador de Datos Artificiales

```
120 library("data.table")
121 library(knitr)
122 library(ggplot2)
123 library(gdata)
124
125 rm(list=ls())
126 setwd("~/Bioinf/Proyecto_Final/artificialHCV/DPmil")
127
128 #Funcion de calculo de sensibilidad
129
130 get_sens <- function(posvector, truepositions) {
131   sens<-(sum(truepositions %in% posvector, na.rm=TRUE)/(length(
132     truepositions)))
133   return(sens)
134 }
135
136 #Funcion de calculo de PPV
137
138 get_ppv<- function(posvector, truepositions) {
139   ppv<-sum(truepositions %in% posvector, na.rm=TRUE)/length(
140     posvector)
141   return(ppv)
142 }
143
144 #Funcion de vector precision: sens-ppv
145
146 get_precision_vector<- function(name, posvector, truepositions,
147   AF) {
148
149   sens<-get_sens(posvector, truepositions)
150   ppv<-get_ppv(posvector, truepositions)
151   F1<-(2*sens*ppv)/(sens+ppv)
152   precision_vector<-data.frame(name, sens, ppv, F1, AF)
153   names(precision_vector)<-c("name", "sens", "ppv", "F1", "AF")

```

```
151   return(precision_vector)
152 }
153
154
155 #Funcion de dataframe de precision sens-ppv
156 get_precision_df<- function(vcfposlist ,truepositions ,AF)
157   {
158     precision_df<-data.frame(name=character() ,sens=numeric() ,ppv
159       =numeric() ,AF=character())
160     for(i in 1:length(vcfposlist)){
161       precision_df<-rbind(precision_df ,get_precision_vector(names(
162         vcfposlist[i] ) ,vcfposlist[[i]] ,truepositions ,AF))
163     }
164     return(precision_df)
165   }
166
167 process_vcfpack<-function ( folder ,AF) {
168
169   setwd( folder )
170   #Extraccion de VCF's
171   lofreqvcf<-readVcf(list.files(pattern="*.lofreq.vcf" , full.names=TRUE))
172   freebayes<-readVcf(list.files(pattern="*.freebayes.vcf" ,
173     full.names=TRUE))
174   gatkvcf<-readVcf(list.files(pattern="*.GATKploidy.vcf" , full.names=TRUE))
175   varscanvcf<-read.table(list.files(pattern="*.varscan.vcf" ,
176     full.names=TRUE))
177   platyvcf<-readVcf(list.files(pattern="*.platy.vcf" , full.names=TRUE))
178   naivevcf<- read.table(list.files(pattern = "*.naivepileup.txt"))
179   varscanvcf<-varscanvcf[-1,]
180
181   #Extraccion de posiciones
```

```
179 lofreqpos<-start(ranges(lofreqvcf))
180 freebayespos<-start(ranges(freebayes))
181 gatkpos<-start(ranges(gatkvcf))
182 platypos<-start(ranges(platyvcf))
183 varscanpos<-as.numeric(as.vector(varscanvcf[,2]))
184 naivepos<-naivevcf$pos
185
186 truepositions<-c
      (3525,3579,3657,3658,3784,3923,3942,6348,6350,6429,6443,6531,6534,
187
188 conditionpositive<-length(truepositions)
189 #Lista de posiciones
190 vcfposlist<-list("lofreq"=lofreqpos,"freebayes"=freebayespos
      ,"gatk"=gatkpos,"platy"=platypos,"varscan"=varscanpos,"
      naive"=naivepos)
191
192 precision_df<-get_precision_df(vcfposlist,truepositions,AF)
193
194 return(precision_df)
195 }
196 precision_df_20pc<-process_vcfpack("~/Bioinf/Proyecto_Final/
      artificialHCV/DPmil/20%","0.2")
197 precision_df_5pc<-process_vcfpack("~/Bioinf/Proyecto_Final/
      artificialHCV/DPmil/5%","0.05")
198 precision_df_1pc<-process_vcfpack("~/Bioinf/Proyecto_Final/
      artificialHCV/DPmil/1%","0.01")
199 precision_df_0.5pc<-process_vcfpack("~/Bioinf/Proyecto_Final/
      artificialHCV/DPmil/0.5%","0.005")
200
201 precision_df<-rbind(precision_df_20pc,precision_df_5pc,
      precision_df_1pc,precision_df_0.5pc)
202
203 # Scatterplot
204 gg1 <- ggplot(precision_df_20pc, aes(x=ppv, y=sens)) +
```

```
205 geom_point(shape=21,col="black",alpha=0.3,aes(fill=name,size
      =name)) +
206
207 xlim(c(0,1)) +
208 ylim(c(0,1)) +
209 scale_fill_hue(l=40, c=65)+
210 labs(subtitle="Scatterplot_{}_Sensibilidad_{}_vs_{}_PPV",
211       y="Sensibilidad",
212       x="PPV",
213       title="Mutaciones_al_20%",
214       caption = "Depth_1000")
215
216 plot(gg1)
217
218 # Scatterplot 5%
219 gg2 <- ggplot(precision_df_5pc, aes(x=ppv, y=sens)) +
220   geom_point(shape=21,col="black",alpha=0.5 ,aes(fill=name,
221     size=name)) +
222   xlim(c(0,1)) +
223   ylim(c(0,1)) +
224   scale_fill_hue(l=40, c=65)+
225   labs(subtitle="Scatterplot_{}_Sensibilidad_{}_vs_{}_PPV",
226         y="Sensibilidad",
227         x="PPV",
228         title="Mutaciones_al_5%",
229         caption = "Depth_1000")
230
231 plot(gg2)
232
233 # Scatterplot 1%
234 gg3 <- ggplot(precision_df_1pc, aes(x=ppv, y=sens)) +
235   geom_point(shape=21,col="black",alpha=0.5 ,aes(fill=name,
236     size=name)) +
237   xlim(c(0,1)) +
238   ylim(c(0,1)) +
239   scale_fill_hue(l=40, c=65)+
```

```
238 labs(subtitle="Scatterplot_ Sensibilidad_vs_PPV",
239       y="Sensibilidad",
240       x="PPV",
241       title="Mutaciones_al_1%",
242       caption = "Depth_1000")
243
244 gg3+scale_color_brewer(palette="BrBG")
245
246 # Scatterplot 0.5%
247 gg4 <- ggplot(precision_df_0.5pc, aes(x=ppv, y=sens)) +
248   geom_point(shape=21,col="black", alpha=0.5 ,aes(fill=name,
249     size=name)) +
250   xlim(c(0,1)) +
251   ylim(c(0,1)) +
252   scale_fill_hue(l=40, c=65)+
253   labs(subtitle="Scatterplot_ Sensibilidad_vs_PPV",
254        y="Sensibilidad",
255        x="PPV",
256        title="Mutaciones_al_0.5%",
257        caption = "Depth_1000")
258
259 plot(gg4)
260
261 sensplot<-ggplot(data=precision_df, aes(x=AF, y=sens ,group=
262   name, colour=name)) +
263   geom_line()+
264   geom_point(aes(shape=name, size=name), size=4, alpha=0.7)+
265   scale_color_hue(l=30, c=65)+
266   # scale_color_brewer(palette="Set1")+
267   scale_shape_manual(values=c(8,7,11,15,10,16))+
268   labs(subtitle="Barplot",
269        y="Sensibilidad",
270        x="AF",
271        title="Sensibilidad",
272        caption = "Depth_1000")
273 sensplot
```

```
272
273 ggplot(data=precision_df, aes(x=AF, y=ppv, group=name, colour=
      name)) +
274   geom_line()+
275   geom_point(aes(shape=name), size=4, alpha=0.7)+
276   scale_color_hue(l=30, c=65)+
277   # scale_color_brewer(palette="Set1")+
278   scale_shape_manual(values=c(8,7,11,15,10,16))+
279   labs(subtitle="Barplot",
280        y="PPV",
281        x="AF",
282        title="PPV",
283        caption = "Depth_1000")
284
285 ggplot(data=precision_df, aes(x=AF, y=F1, group=name, colour=
      name)) +
286   geom_line()+
287   geom_point(aes(shape=name), size=4, alpha=0.7)+
288   scale_color_hue(l=30, c=65)+
289   # scale_color_brewer(palette="Set1")+
290   scale_shape_manual(values=c(8,7,11,15,10,16))+
291   labs(subtitle="Barplot",
292        y="F1",
293        x="AF",
294        title="F1",
295        caption = "Depth_1000")
```

### B.3. Evaluador de Datos Reales

```

297 library("data.table")
298 library(knitr)
299 library(ggplot2)
300
301 rm(list=ls())
302 setwd("~/Bioinf/realseq")
303
304 # Sensibilidad = VP/ CP (Verdaderos Positivos/Condicionalmnete
      positivos)
305
306 # PPV = VP / PP (Verdaderos Positivos/ Predicciones positivas)
307
308 #Funciones de calculo de sens y PPV
309
310 get_VP<-function(posvector, truepositions) {
311   uniques<-sum(unique(truepositions) %in% unique(posvector), na.
      rm=TRUE)
312   duplicated1<-sum(unique(truepositions[ duplicated(
      truepositions) ])%in% unique(posvector[ duplicated(
      posvector) ]), na.rm=TRUE)
313   duplicated2<-sum(truepositions[ duplicated(truepositions) ][
      duplicated(truepositions[ duplicated(truepositions) ])]%in%
      posvector[ duplicated(posvector) ] [ duplicated(posvector[
      duplicated(posvector) ])], na.rm=TRUE)
314   VP<-uniques+duplicated1+duplicated2
315   return(VP)
316 }
317
318 get_CP<-function(truepositions) {
319   CP<-length(truepositions)
320   return(CP)
321 }
322
323 get_PP<-function(posvector, filtered_naivepos) {

```

```

324  uniques<-sum(unique(filtered_naivepos) %in% unique(posvector) ,
      na.rm=TRUE)
325  duplicated1<-sum(unique(filtered_naivepos[duplicated(
      filtered_naivepos)]) %in% unique(posvector[duplicated(
      posvector)]), na.rm=TRUE)
326  duplicated2<-sum(filtered_naivepos[duplicated(filtered_
      naivepos)][duplicated(filtered_naivepos[duplicated(
      filtered_naivepos)])] %in% posvector[duplicated(posvector)
      ][duplicated(posvector[duplicated(posvector)])], na.rm=
      TRUE)
327  PP<-uniques+duplicated1+duplicated2
328  return(PP)
329  }
330
331  #Funcion de vector : VP,PP,CP
332  get_precision_vector<- function(posvector, truepositions,
      filtered_naivepos){
333
334  VP<-get_VP(posvector, truepositions)
335  PP<-get_PP(posvector, filtered_naivepos)
336  CP<-get_CP(truepositions)
337  precision_vector<-data.frame(VP,PP,CP)
338  names(precision_vector)<-c("VP", "PP", "CP")
339  return(precision_vector)
340  }
341
342  #Funcion de dataframe de precision sens-ppv
343  get_precision_df<- function(vcfposlist, truepositions, filtered_
      naivepos){
344  precision_df<-data.frame(VP=numeric(), PP=numeric(), CP=
      numeric())
345  for(i in 1:length(vcfposlist)){
346    precision_df<-rbind(precision_df, get_precision_vector(
      vcfposlist[[i]], truepositions, filtered_naivepos))
347  }
348  return(precision_df)

```

```
349 }
350
351
352 get_allnaivepos<-function(vcf){
353   positions<-vcf$pos
354
355   pos_underfreq<-numeric()
356
357   for(i in 1:length(positions)){
358     #ALT DNA letter at position:
359     ALT_posnumber<-sum((vcf[i,3:6])!=0, na.rm=TRUE)
360
361
362     for(j in 1:ALT_posnumber){
363
364       AF_pos=positions[i] #posiciones alteradas
365
366       pos_underfreq<-c(pos_underfreq,AF_pos)
367     }
368   }
369   return(pos_underfreq)
370 }
371
372 #Funcion que filtra posiciones a un umbral de frecuencias
373 freq_filter<-function(vcf, freq){
374   positions<-vcf$pos
375
376
377   pos_underfreq<-numeric()
378
379   for(i in 1:length(positions)){
380     #ALT DNA letter at position:
381     ALT_posnumber<-sum((vcf[i,3:6])!=0, na.rm=TRUE)
382     ALT_coordenadas <-which((vcf[i,3:6])!=0)
383
384     for(j in 1:ALT_posnumber){
```

```

385
386     AF=vcf[i,ALT_coordenadas[j]+2]
387     AF_pos=positions[i] #posiciones alteradas
388
389     if(AF<=freq){
390         pos_underfreq<-c(pos_underfreq,AF_pos)
391     }
392 }
393 }
394
395 return(pos_underfreq)
396 }
397
398 #Funcion que obtiene todas las posiciones naive que provocan
      cambios de AA
399
400 get_truepos<-function(vcf, gene, gene_startpos, freq_th){
401     vcf<-vcf[vcf$pos>gene_startpos,]
402     positions<-vcf$pos
403     Achange_vector<-numeric()
404     posAchange_vector<-numeric()
405     bases<-c("A", "T", "C", "G")
406     prot<-translate(gene)[[1]]
407
408     for(i in 1:length(positions)){
409         #ALT DNA letter at position:
410
411         ALT_posnumber<-sum((vcf[i,3:6])!=0, na.rm=TRUE)
412         ALT_coordenadas <-which((vcf[i,3:6])!=0)
413         for(j in 1:ALT_posnumber){
414             if(vcf[i,ALT_coordenadas[j]+2]<freq_th){
415                 ALT_letter=bases[ALT_coordenadas[j]]
416                 ALT_pos=positions[i] #posiciones alteradas
417
418                 ALT_NS5bpos=ALT_pos-(gene_startpos-1) #posicion 0 de
                        NS5b

```

```
419
420     ALTprot<-translate(replaceLetterAt(gene[[1]], ALT_
421       NS5bpos, ALT_letter))
422     m<-matchPattern(ALTprot, prot, max.mismatch = 1, fixed
423       = TRUE)
424     altAApos<-unlist(mismatch(ALTprot, m))
425     refAA<-as.character(prot[altAApos])
426     altAA<-as.character(ALTprot[altAApos])
427
428     if (paste(refAA, altAApos, altAA, sep="")==") {
429       AChange_vector<-c(AChange_vector, "Silent")
430       posAChange_vector<-c(posAChange_vector, 0)
431     }
432     else {
433       AChange_vector<-c(AChange_vector, paste(refAA,
434         altAApos, altAA, sep=""))
435       posAChange_vector<-c(posAChange_vector, ALT_pos)
436     }
437   }
438 }
439
440 posAChange_vector<- posAChange_vector[posAChange_vector!=
441   0]
442 return(posAChange_vector)
443 }
444
445 #Funcion que devuelve matrices de sensibilidad y ppv
446
447 process_vcfpack<-function(bam_number, truepositions, freq_th,
448   protein = 5){
449
450   if (protein==3){
451     setwd("~/Bioinf/2realseq")
452   }
453   if (protein==5){
454     setwd("~/Bioinf/realseq")
455   }
456 }
```

```
450 }
451
452 #Extraccion de filenames
453 lofreqfilenames<-list.files(pattern="*.lofreq.vcf", full.
      names=TRUE)
454 freebayesfilenames<-list.files(pattern="*.freebayes.vcf",
      full.names=TRUE)
455 gatkvcffilenames<-list.files(pattern="*.GATKploidy.vcf",
      full.names=TRUE)
456 varscanvcffilenames<-list.files(pattern="*.varscan.vcf",
      full.names=TRUE)
457 platyvcaffilenames<-list.files(pattern="*.platy.vcf", full.
      names=TRUE)
458 naivevcffilenames<-list.files(pattern = "*.naivecalls.txt")
459
460 #Extraccion de VCF's
461 lofreqvcf<-readVcf(lofreqfilenames[bam_number])
462 freebayes<-readVcf(freebayesfilenames[bam_number])
463 gatkvcf<-readVcf(gatkvcffilenames[bam_number])
464 varscanvcf<-read.table(varscanvcffilenames[bam_number])
465 platyvvcf<-readVcf(platyvcffilenames[bam_number])
466 naivevcf<-read.table(naivevcffilenames[bam_number])
467 varscanvcf<-varscanvcf[-1,]
468
469   if(protein==3){
470     naivevcf<-naivevcf[naivevcf$pos<5000,]
471   }
472
473 #Extraccion de posiciones
474
475 lofreqpos<-start(ranges(lofreqvcf))
476 freebayespos<-start(ranges(freebayes))
477 gatkpos<-start(ranges(gatkvcf))
478 platypos<-start(ranges(platyvcf))
479 varscanpos<-as.numeric(as.vector(varscanvcf[,2]))
480
```

```
481 naivepos<-get_allnaivepos (naivevcf)
482
483
484
485 filtered_naivepos<-freq_filter (naivevcf , freq_th)
486
487
488
489 #Lista de posiciones
490 vcfposlist<-list ("lofreq"=lofreqpos , "freebayes"=freebayespos
    , "gatk"=gatkpos , "platy"=platypos , "varscan"=varscanpos ,
    naive"=naivepos)
491
492 precision_df<-get_precision_df(vcfposlist , truepositions ,
    filtered_naivepos)
493 return (precision_df)
494 }
495
496 process_batch<-function (bam_sum, gettruepos , freq_th , protein) {
497
498 precision_df<-data.frame (VP=numeric (6) ,PP=numeric (6) ,CP=
    numeric (6))
499 if (protein==3){
500 setwd ("~/Bioinf/2realseq")
501 }
502 if (protein==5){
503 setwd ("~/Bioinf/realseq")
504 }
505 naivevcffilenames<-list.files (pattern = "*.naivecalls.txt"
    )
506 naivevcf<- read.table (naivevcffilenames [15])
507 filtered_naivepos<-freq_filter (naivevcf , freq_th)
508
509 #Get true positions
510 if (gettruepos==1){
511 if (protein==3){
```

```

512     gene<- readDNAStringSet("NS3.fasta", format="fasta", nrec
513         =-1L, skip=0L, seek.first.rec=FALSE, use.names=TRUE)
514     gene_startpos<-3420
515     }
516     if (protein==5){
517         gene<- readDNAStringSet("NS5b.fasta", format="fasta",
518             nrec=-1L, skip=0L, seek.first.rec=FALSE, use.names=
519             TRUE)
520         gene_startpos<-7602
521     }
522     truepositions<-get_truepos (naivevcf , gene , gene_startpos ,
523         freq_th)
524 }
525 else {
526     if (protein==3){
527         truedocumentedpositions<-c
528             (3525,3579,3657,3658,3784,3923,3942)
529     }
530     if (protein==5){
531         truedocumentedpositions<-c
532             (7842,8073,8074,8075,8545,8840,8932,8941,8950,9256,9264,9266,9
533     }
534
535     truepositions<-truedocumentedpositions [
536         truedocumentedpositions %n %filtered_naivepos ]
537 }
538 for (i in 1:bam_sum){
539     precision_df<-precision_df+process_vcfpack (i , truepositions
540         , freq_th , protein)
541     print ("un_bam_mas")
542 }
543 AF<-rep ( freq_th*0.5 ,6)
544 precision_df<-cbind (precision_df ,AF)
545 return (precision_df)
546 }

```

```
539 get_stats<-function (df1 , df2) {
540   df<-df1+df2
541   sens<-df$VP/df$CP
542   PPV<-df$VP/df$PP
543   F1<-(2*sens*PPV)*1000/(sens+PPV)
544   df$AF<-as.character(df$AF)
545   name<-c("lofreq","freebayes","gatk","platy","varscan","naive
546     ")
547   df<-cbind(name,df,sens, PPV,F1)
548
549   return(df)
550 }
551 df_ns3_100pc<-process_batch(16,gettruepos = 1,1,3)
552 df_ns5b_100pc<-process_batch(16,gettruepos = 1,1,5)
553 df_100pc<-get_stats(df_ns5b_100pc,df_ns3_100pc)
554
555 # Scatterplot 100%
556 gg1 <- ggplot(df_100pc, aes(x=PPV, y=sens)) +
557   geom_point(shape=21,col="black",alpha=0.8,aes(fill=name,size
558     =name)) +
559
560   xlim(c(0,0.6)) +
561   ylim(c(0,1)) +
562   scale_fill_hue(l=50, c=85)+
563   labs(subtitle="Scatterplot_{}_{}_Sensibilidad_{}_{}_PPV",
564     y="Sensibilidad",
565     x="PPV",
566     title="Mutaciones_al_100%",
567     caption = "NS3_y_NS5B")
568
569 plot(gg1)
570
571 df_ns5b_20pc<-process_batch(16,gettruepos = 1,0.2,5)
572 df_ns3_20pc<-process_batch(16,gettruepos = 1,0.2,3)
573 df_20pc<-get_stats(df_ns5b_20pc,df_ns3_20pc)
```

```
573 # Scatterplot 20%
574 gg1 <- ggplot(df_20pc, aes(x=PPV, y=sens)) +
575   geom_point(shape=21,col="black",alpha=0.8,aes(fill=name,size
576     =name)) +
577   xlim(c(0,0.6)) +
578   ylim(c(0,1)) +
579   scale_fill_hue(l=50, c=85)+
580   labs(subtitle="Scatterplot_ Sensibilidad_vs._PPV",
581     y="Sensibilidad",
582     x="PPV",
583     title="Mutaciones_al_20%",
584     caption = "NS3_y_NS5B")
585
586 plot(gg1)
587
588 df_ns3_5pc<-process_batch(16,gettruepos = 1,0.05,3)
589 df_ns5b_5pc<-process_batch(16,gettruepos = 1,0.05,5)
590 df_5pc<-get_stats(df_ns5b_5pc,df_ns3_5pc)
591
592 # Scatterplot 5%
593 gg1 <- ggplot(df_5pc, aes(x=PPV, y=sens)) +
594   geom_point(shape=21,col="black",alpha=0.8,aes(fill=name,size
595     =name)) +
596   xlim(c(0,0.6)) +
597   ylim(c(0,1)) +
598   scale_fill_hue(l=50, c=85)+
599   labs(subtitle="Scatterplot_ Sensibilidad_vs._PPV",
600     y="Sensibilidad",
601     x="PPV",
602     title="Mutaciones_al_5%",
603     caption = "NS3_y_NS5B")
604
605 plot(gg1)
606
```

```
607 df_ns3_1pc<-process_batch(16, gettruepos = 1,0.01,3)
608 df_ns5b_1pc<-process_batch(16, gettruepos = 1,0.01,5)
609 df_1pc<-get_stats(df_ns5b_1pc,df_ns3_1pc)
610
611 # Scatterplot 1%
612 gg1 <- ggplot(df_1pc, aes(x=PPV, y=sens)) +
613   geom_point(shape=21,col="black",alpha=0.8,aes(fill=name,size
614     =name)) +
615   xlim(c(0,0.6)) +
616   ylim(c(0,1)) +
617   scale_fill_hue(l=50, c=85)+
618   labs(subtitle="Scatterplot_1_Sensibilidad_vs_PPV",
619     y="Sensibilidad",
620     x="PPV",
621     title="Mutaciones_al_1%",
622     caption = "NS3_y_NS5B")
623
624 plot(gg1)
625
626 df_ns3_0.5pc<-process_batch(16, gettruepos = 1,0.005,3)
627 df_ns5b_0.5pc<-process_batch(16, gettruepos = 1,0.005,5)
628 df_0.5pc<-get_stats(df_ns5b_0.5pc,df_ns3_0.5pc)
629
630 # Scatterplot 0.5%
631 gg1 <- ggplot(df_0.5pc, aes(x=PPV, y=sens)) +
632   geom_point(shape=21,col="black",alpha=0.8,aes(fill=name,size
633     =name)) +
634   xlim(c(0,0.6)) +
635   ylim(c(0,1)) +
636   scale_fill_hue(l=50, c=85)+
637   labs(subtitle="Scatterplot_0.5_Sensibilidad_vs_PPV",
638     y="Sensibilidad",
639     x="PPV",
640     title="Mutaciones_al_0.5%",
```

```
641     caption = "NS3_y_NS5B")
642
643 plot(gg1)
644
645 precision_df<-rbind(df_0.5pc, df_1pc,df_5pc,df_20pc,df_100pc)
646
647
648 sensplot<-ggplot(data=precision_df, aes(x=AF, y=sens,group=
649     name,colour=name)) +
650     geom_line()+
651     geom_point(aes(shape=name, size=name), size=4,alpha=0.7)+
652     scale_color_hue(l=30, c=65)+
653     # scale_color_brewer(palette="Set1")+
654     scale_x_discrete(limits = unique(rev(precision_df$AF)))+
655     scale_shape_manual(values=c(8,7,11,15,10,16))+
656     labs(subtitle="Barplot",
657         y="Sensibilidad",
658         x="AF",
659         title="Sensibilidad",
660         caption = "NS3_y_NS5B")
661 sensplot
662
663 ggplot(data=precision_df, aes(x=AF, y=PPV,group=name,colour=
664     name)) +
665     geom_line()+
666     geom_point(aes(shape=name), size=4,alpha=0.7)+
667     scale_color_hue(l=30, c=65)+
668     # scale_color_brewer(palette="Set1")+
669     scale_x_discrete(limits = unique(rev(precision_df$AF)))+
670     scale_shape_manual(values=c(8,7,11,15,10,16))+
671     labs(subtitle="Barplot",
672         y="PPV",
673         x="AF",
674         title="PPV",
675         caption = "NS3_y_NS5B")
```

```
675  
676  
677 ggplot(data=precision_df, aes(x=AF, y=F1, group=name, colour=  
678   name)) +  
679   geom_line() +  
680   geom_point(aes(shape=name), size=4, alpha=0.7) +  
681   scale_color_hue(l=30, c=65) +  
682   scale_x_discrete(limits = unique(rev(precision_df$AF))) +  
683   # scale_color_brewer(palette="Set1") +  
684   scale_shape_manual(values=c(8,7,11,15,10,16)) +  
685   labs(subtitle="Barplot",  
686        y="F1",  
687        x="AF",  
688        title="F1_score",  
689        caption = "NS3_y_NS5B")
```

# Bibliografía

- [1] N. R. Council, "Toward precision medicine: Building a knowledge network for biomedical research and a new taxonomy of disease," *Washington, DC: The National Academies Press*, 2011.
- [2] E. Ashley, "Towards precision medicine," *Nat Rev Genet*, vol. 17, no. 9, pp. 507–522, 2016.
- [3] H. Hackl, P. Charoentong, F. Finotello, and Z. Trajanoski, "Computational genomics tools for dissecting tumour-immune cell interactions," *Nat Rev Genet*, vol. 17, no. 8, p. nrg.2016.67, 2016.
- [4] K. E. Caudle, T. E. Klein, J. M. Hoffman, D. J. Muller, W. Michelle, L. Gong, M. E. M. K. Sangkuhl, C. F. Thorn, M. Schwab, J. A. Agundez, R. R. Freimuth, V. Huser, M. T. Lee, O. F. Iwuchukwu, K. R. Crews, S. A. Scott, M. Wadelius, J. J. Swen, R. F. Tyndale, C. Stein, D. Roden, M. V. Relling, M. S. Williams, and S. G. Johnson, "Incorporation of pharmacogenomics into routine clinical practice: the clinical pharmacogenetics implementation consortium (CPIC) guideline development process." *Curr. Drug Metab.*, vol. 15, no. 2, pp. 209–17, 2014.
- [5] F. Collins and M. Victor, "Implications of the human genome project for medical science," *Jama*, vol. 285, no. 5, pp. 540–544, 2001.
- [6] I. Ochoa, M. Hernaez, R. Goldfeder, T. Weissman, and E. Ashley, "Effect of lossy compression of quality scores on variant calling," *Brief Bioinform*, p. bbw011, 2016.
- [7] R. Goldfeder, J. Priest, J. Zook, M. Grove, D. Waggott, M. Wheeler, M. Salit, and E. Ashley, "Medical implications of technical accuracy in genome sequencing," *Genome Medicine*, vol. 8, no. 1, p. 24, 2016.
- [8] M. D., T. Manolio, D. Dimmock, H. Rehm, J. Shendure, G. Abecasis, D. Adams, R. Altman, S. Antonarakis, E. Ashley, J. Barrett, L. Biesecker, D. Conrad, G. Cooper, N. Cox, M. Daly, M. Gerstein, D. Goldstein, J. Hirschhorn, S. Leal, L. Pennac-

- chio, J. Stamatoyannopoulos, S. Sunyaev, D. Valle, B. Voight, W. Winckler, and C. Gunter, "Guidelines for investigating causality of sequence variants in human disease," *Nature*, vol. 508, no. 7497, p. 469, 2014.
- [9] B. Lin, J. Wang, and Y. Cheng, "Recent patents and advances in the Next-Generation sequencing technologies." *Recent Pat Biomed Eng*, vol. 2008, no. 1, pp. 60–67, 2008.
- [10] S. Goodwin, M. J. D, and M. R. W, "Coming of age: ten years of next-generation sequencing technologies," *Nat Rev Genet*, vol. 17, no. 6, pp. 333–351, 2016.
- [11] U. Landegren, R. Kaiser, J. Sanders, and L. Hood, "A ligase-mediated gene detection technique." *Science*, vol. 241, no. 4869, pp. 1077–80, 1988.
- [12] M. Margulies, M. Egholm, W. Altman, S. Attiya, J. Bader, L. Bemben, J. Berka, M. Braverman, Y. Chen, Z. Chen, S. Dewell, L. Du, J. Fierro, X. Gomes, B. Godwin, W. He, S. Helgesen, C. Ho, G. Irzyk, S. Jando, M. Alenquer, T. Jarvie, K. Jirage, J. Kim, J. Knight, J. Lanza, J. Leamon, S. Lefkowitz, M. Lei, J. Li, K. Lohman, H. Lu, V. Makhijani, M. Keith, M. Michael, E. Myers, E. Nickerson, J. Nobile, R. Plant, B. Puc, M. Ronan, G. Roth, G. Sarkis, J. Simons, J. Simpson, M. Srinivasan, K. Tartaro, A. Tomasz, K. Vogt, G. Volkmer, S. Wang, Y. Wang, M. Weiner, P. Yu, R. Begley, and J. Rothberg, "Genome sequencing in microfabricated high-density picolitre reactors," *Nature*, vol. 437, no. 7057, p. 376, 2005.
- [13] J. Rothberg, W. Hinz, T. Rearick, J. Schultz, W. Mileski, M. Davey, J. Leamon, K. Johnson, M. Milgrew, M. Edwards, J. Hoon, J. Simons, D. Marran, J. Myers, J. Davidson, A. Branting, J. Nobile, B. Puc, D. Light, T. Clark, M. Huber, J. Branciforte, I. Stoner, S. Cawley, M. Lyons, Y. Fu, N. Homer, M. Sedova, X. Miao, B. Reed, J. Sabina, E. Feierstein, M. Schorn, M. Alanjary, E. Dimalanta, D. Dressman, R. Kasinskas, T. Sokolsky, J. Fidanza, E. Namsaraev, M. Kevin, A. Williams, G. Roth, and J. Bustillo, "An integrated semiconductor device enabling non-optical genome sequencing," *Nature*, vol. 475, no. 7356, p. 348, 2011.
- [14] J. Eid, A. Fehr, J. Gray, K. Luong, J. Lyle, G. Otto, P. Peluso, D. Rank, P. Baybayan, B. Bettman, A. Bibillo, K. Bjornson, B. Chaudhuri, F. Christians, R. Cicero, S. Clark, R. Dalal, A. Dewinter, J. Dixon, M. Foquet, A. Gaertner, P. Hardenbol, C. Heiner, K. Hester, D. Holden, G. Kearns, X. Kong, R. Kuse, Y. Lacroix, S. Lin, P. Lundquist, C. Ma, P. Marks, M. Maxham, D. Murphy, I. Park, T. Pham, M. Phillips, J. Roy, R. Sebra, G. Shen, J. Sorenson, A. Tomaney, K. Travers, M. Trulson, J. Vieceli, J. We-

- gener, D. Wu, A. Yang, D. Zaccarin, P. Zhao, F. Zhong, J. Korlach, and S. Turner, "Real-time DNA sequencing from single polymerase molecules." *Science*, vol. 323, no. 5910, pp. 133–8, 2009.
- [15] J. Clarke, H. Wu, L. Jayasinghe, A. Patel, S. Reid, and H. Bayley, "Continuous base identification for single-molecule nanopore DNA sequencing," *Nat Nanotechnol*, vol. 4, no. 4, p. nnano.2009.12, 2009.
- [16] C. Knief, "Analysis of plant microbe interactions in the era of next generation sequencing technologies," *Frontiers Plant Sci*, vol. 5, p. 216, 2014.
- [17] B. Merriman, I. Team, and J. Rothberg, "Progress in ion torrent semiconductor chip based sequencing," *Electrophoresis*, vol. 33, no. 23, pp. 3397–3417, 2012.
- [18] P. Holland, R. Abramson, R. Watson, and D. Gelfand, "Detection of specific polymerase chain reaction product by utilizing the 5'—3' exonuclease activity of thermus aquaticus DNA polymerase." *Proc National Acad Sci*, vol. 88, no. 16, pp. 7276–7280, 1991.
- [19] V. Heather, K. Vrana, and W. Freeman, "Twenty-five years of quantitative PCR for gene expression analysis," *Biotechniques*, vol. 44 Supplement, no. 4, pp. 619–626, 2008.
- [20] L. Henderson, C. Applegate, E. Wohler, M. Sheridan, H. Julie, and D. Batista, "The impact of chromosomal microarray on clinical management: a retrospective analysis," *Genet Med*, vol. 16, no. 9, p. 657, 2014.
- [21] S. Zhao, F. Wai-Ping, A. Bittner, K. Ngo, and X. Liu, "Comparison of RNA-Seq and microarray in transcriptome profiling of activated t cells," *Plos One*, vol. 9, no. 1, p. e78644, 2014.
- [22] G. J. Tsongalis, E. Chao, J. M. Hagenkord, T. Hambuch, and J. H. Moore, "Bioinformatics: What the clinical laboratorian needs to know and prepare for," *Clin Chem*, vol. 59, no. 9, pp. 1301–1305, 2013.
- [23] G. Oliver, S. Hart, and E. Klee, "Bioinformatics for clinical next generation sequencing," *Clin Chem*, vol. 61, no. 1, pp. 124–135, 2015.
- [24] T. Subazini, P. Sen, and I. Nookaew, "Evaluation and assessment of read-mapping by multiple next-generation sequencing aligners based on genome-wide characteristics," *Genomics*, vol. 109, no. 3-4, pp. 186–191, 2017.

- [25] H. Li, "Aligning sequence reads, clone sequences and assembly contigs with bwa-mem," *Cornell University Library*, no. arXiv:1303.3997, 2013.
- [26] H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, and R. Durbin, "The sequence Alignment/Map format and SAMtools," *Bioinformatics*, vol. 25, no. 16, pp. 2078–2079, 2009.
- [27] S. Sandmann, A. Graaf, M. Karimi, B. Reijden, H. Eva, J. Jansen, and M. Dugas, "Evaluating variant calling tools for Non-Matched Next-Generation sequencing data," *Sci Reports*, vol. 7, p. 43169, 2017.
- [28] M. Mielczarek and J. Szyda, "Review of alignment and SNP calling algorithms for next-generation sequencing data," *J Appl Genet*, vol. 57, no. 1, pp. 71–79, 2016.
- [29] N. D. Roberts, D. R. Kortschak, W. T. Parker, A. W. Schreiber, S. Branford, H. S. Scott, G. Glonek, and D. L. Adelson, "A comparative analysis of algorithms for somatic SNV detection in cancer," *Bioinformatics*, vol. 29, no. 18, pp. 2223–2230, 2013.
- [30] R. Nielsen, J. S. Paul, A. Albrechtsen, and Y. S. Song, "Genotype and SNP calling from next-generation sequencing data," *Nat Rev Genet*, vol. 12, no. 6, p. 443, 2011.
- [31] C. Xu, "A review of somatic single nucleotide variant calling algorithms for next-generation sequencing data," *Comput Struct Biotechnology J*, 2018.
- [32] J. Wang, T. Skoog, E. Einarsson, T. Kaartokallio, H. Laivuori, A. Grauers, P. Gerdhem, M. Hytönen, H. Lohi, J. Kere, and H. Jiao, "Investigation of rare and low-frequency variants using high-throughput sequencing with pooled DNA samples," *Sci Reports*, vol. 6, no. 1, p. 33256, 2016.
- [33] A. Wilm, P. Aw, D. Bertrand, G. Yeo, S. Ong, C. Wong, C. Khor, R. Petric, M. Hibberd, and N. Nagarajan, "LoFreq: a sequence-quality aware, ultra-sensitive variant caller for uncovering cell-population heterogeneity from high-throughput sequencing datasets," *Nucleic Acids Res*, vol. 40, no. 22, pp. 11 189–11 201, 2012.
- [34] D. Koboldt, Q. Zhang, D. Larson, D. Shen, M. Michael, L. Lin, C. Miller, E. Mardis, L. Ding, and R. Wilson, "VarScan 2: Somatic mutation and copy number alteration discovery in cancer by exome sequencing," *Biotechfor*, vol. 22, no. 3, pp. 568–576, 2012.
- [35] D. Mark, E. Banks, R. Poplin, K. Garimella, J. Maguire, C. Hartl, A. Philippakis,

- G. Angel, M. Rivas, M. Hanna, M. Aaron, T. Fennell, A. Kernytsky, A. Sivachenko, K. Cibulskis, S. Gabriel, D. Altshuler, and M. Daly, "A framework for variation discovery and genotyping using next-generation DNA sequencing data," *Nat Genet*, vol. 43, no. 5, pp. 491–498, 2011.
- [36] A. Rimmer, H. Phan, I. Mathieson, Z. Iqbal, S. Twigg, W. Consortium, A. Wilkie, M. Gil, and G. Lunter, "Integrating mapping-, assembly- and haplotype-based approaches for calling variants in clinical sequencing applications," *Nat Genet*, vol. 46, no. 8, pp. 912–918, 2014.
- [37] G. Garrison, E. & Marth, "Haplotype-based variant detection from short-read sequencing," *Cornell University Library*, no. arXiv:1207.3907, 2012.
- [38] T. Kish, A. Aziz, and M. Sorio, "Hepatitis c in a new era: A review of current therapies." *P T*, vol. 42, no. 5, pp. 316–329, 2017.
- [39] P. Angeleri, M. de los Angeles, J. Solari, and G. Vidiella, "Las hepatitis virales en la argentina," *Ministerio de Salud de la Nación Argentina*, 2014.
- [40] R. Francesco, "Molecular virology of the hepatitis c virus," *J Hepatol*, vol. 31, pp. 47–53, 1999.
- [41] T. Scheel and C. Rice, "Understanding the hepatitis c virus life cycle paves the way for highly effective therapies," *Nat Med*, vol. 19, no. 7, pp. 837–849, 2013.
- [42] M. Hijikata, H. Mizushima, Y. Tanji, Y. Komoda, Y. Hirowatari, T. Akagi, N. Kato, K. Kimura, and K. Shimotohno, "Proteolytic processing and membrane association of putative nonstructural proteins of hepatitis c virus," *Proc National Acad Sci*, vol. 90, no. 22, pp. 10 773–10 777, 1993.
- [43] S. Fourati and J. Pawlotsky, "Virologic tools for HCV drug resistance testing," *Viruses*, vol. 7, no. 12, pp. 6346–6359, 2015.
- [44] S. Nitta, Y. Asahina, M. Matsuda, N. Yamada, R. Sugiyama, T. Masaki, R. Suzuki, N. Kato, M. Watanabe, T. Wakita, and T. Kato, "Effects of Resistance-Associated NS5A mutations in hepatitis c virus on viral production and susceptibility to antiviral reagents," *Sci Reports*, vol. 6, no. 1, p. srep34652, 2016.
- [45] A. Ahmed and D. Felmlee, "Mechanisms of hepatitis c viral resistance to direct acting antivirals," *Viruses*, vol. 7, no. 12, pp. 6716–6729, 2015.
- [46] M. Ogishi, H. Yotsuyanagi, T. Tsutsumi, H. Gatanaga, H. Ode, W. Sugiura, K. Mo-

- riya, S. Oka, S. Kimura, and K. Koike, "Deconvoluting the composition of Low-Frequency hepatitis c viral quasispecies: Comparison of genotypes and NS3 Resistance-Associated variants between HCV/HIV coinfecting hemophiliacs and HCV mono-infected patients in Japan," *Plos One*, vol. 10, no. 3, p. e0119145, 2015.
- [47] Z.-w. Chen, H. Li, H. Ren, and P. Hu, "Global prevalence of pre-existing HCV variants resistant to direct-acting antiviral agents (DAAs): mining the GenBank HCV genome data," *Sci Reports*, vol. 6, no. 1, p. srep20310, 2016.
- [48] N. Marascio, G. Pavia, A. Strazzulla, T. Dierckx, L. Cuypers, B. Vrancken, G. Barreca, T. Mirante, D. Malanga, D. Oliveira, A. Vandamme, C. Torti, M. Liberto, and A. Focà, "Detection of natural Resistance-Associated substitutions by ion semiconductor technology in HCV1b positive, Direct-Acting antiviral Agents-Naïve patients," *Int J Mol Sci*, vol. 17, no. 9, p. 1416, 2016.
- [49] B. Wei, J. Kang, M. Kibukawa, L. Chen, P. Qiu, F. Lahser, M. Marton, and D. Levitan, "Development and validation of a Template-Independent Next-Generation sequencing assay for detecting Low-Level Resistance-Associated variants of hepatitis c virus," *J Mol Diagnostics*, vol. 18, no. 5, pp. 643–656, 2016.
- [50] B. Verbist, K. Thys, J. Reumers, Y. Wetzels, K. der Borght, W. Talloen, J. Aerssens, L. Clement, and O. Thas, "VirVarSeq: a low-frequency virus variant detection pipeline for illumina sequencing using adaptive base-calling accuracy filtering," *Bioinformatics*, vol. 31, no. 1, pp. 94–101, 2015.
- [51] S. J. Watson, M. R. Welkers, D. P. Depledge, E. Coulter, J. M. Breuer, M. D. de Jong, and P. Kellam, "Viral population analysis and minority-variant detection using short read next-generation sequencing," *Phil Trans R Soc B*, vol. 368, no. 1614, p. 20120205, 2013.
- [52] A. Beloukas, S. King, K. Childs, A. Papadimitropoulos, M. Hopkins, M. Atkins, K. Agarwal, M. Nelson, and A. Geretti, "Detection of the NS3 Q80K polymorphism by sanger and deep sequencing in hepatitis c virus genotype 1a strains in the UK," *Clin Microbiol Infect*, vol. 21, no. 11, pp. 1033–1039, 2015.
- [53] D. Kliemann, C. Tovo, A. Veiga, A. Mattos, and C. Wood, "Polymorphisms and resistance mutations of hepatitis c virus on sequences in the European hepatitis c virus database," *World J Gastroenterol*, vol. 22, no. 40, p. 8910, 2016.
- [54] X. Yang, P. Charlebois, A. Macalalad, M. Henn, and M. Zody, "V-Phaser 2: variant

inference for viral populations," *Bmc Genomics*, vol. 14, no. 1, pp. 1–10, 2013.

- [55] P. Susana, D. Seifert, and N. Beerenwinkel, "Recent advances in inferring viral diversity from high-throughput sequencing data," *Virus Res*, vol. 239, pp. 17–32, 2017.
- [56] S. Bartlett, J. Grebely, A. Eltahla, J. Reeves, A. Howe, V. Miller, C. Francesca, R. Bull, M. Douglas, G. Dore, P. Harrington, A. Lloyd, B. Jacka, G. Matthews, G. Wang, J. Pawlotsky, J. Feld, J. Schinkel, F. Garcia, J. Lennerstrand, and T. Applegate, "Sequencing of hepatitis c virus for detection of resistance to direct-acting antiviral therapy: A systematic review," *Hepatology Commun*, vol. 1, no. 5, pp. 379–390, 2017.
- [57] M. Frampton and R. Houlston, "Generation of artificial FASTQ files to evaluate the performance of Next-Generation sequencing pipelines," *Plos One*, vol. 7, no. 11, p. e49110, 2012.

# Índice de figuras

1.1. Procedimientos de preparación de muestras para plataformas NGS. <i>Knief 2014</i> . . . . .	6
1.2. Escalamiento de Chips. (A) Nombre del Chip. (B) Número de microcubetas y sensores accesibles con el total en paréntesis. (C) Imágen con vista superior del chip. (D) Tamaño relativo del area de sensores en el chip. (E) Secciones transversales vista por microscopía electrónica a través del arreglo de sensores (microcubetas individuales apreciables). (F) La forma de función de sensado con cada Chip. <i>Merriman et al. 2012</i>	8
1.3. Dispositivo de sensor de pH semiconductor y su funcionamiento (A) Configuración de microesfera en un <i>microwell</i> con sensor pHFET.(B) Las señales de pH son muestreadas a una alta frecuencia (curva azul) y luego obteniendo una figura filtrada (curva roja). (C) Las señales de incorporación para cada flujo, mostrando la ausencia de incorporaciones, e incorporaciones de 1, 2 y más bases seguidas. Esta señal es el fundamento para el llamado de bases. <i>Merriman et al. 2012</i> . . . . .	9
1.4. Etapas del análisis bioinformático <i>Oliver et al. 2015</i> . . . . .	12
1.5. Genoma del VHC (arriba) y procesamiento de poliproteína (abajo). Los triángulos representan puntos de corte peptídico, los puntos negros sitios de glicosilación de proteínas. <i>Scheel et al. 2013</i> . . . . .	18
1.6. Ciclo de vida del VHC: 1.Circulación de partícula viral. 2. Entrada a la célula huésped e interacción con el receptor. 3. Traducción y procesamiento de la poliproteína. 4. Replicación de ARN viral. 5. Ensamblaje y morfogénesis del virión. <i>Scheel et al. 2013</i> . . . . .	19

4.1. Visualización de posiciones de variantes detectadas por cada variant caller. Se visualizan los casos para el mayor y el menor AF (20 % y 0,05 %, respectivamente) . . . . .	37
4.2. Diagrama de dispersión de sensibilidad vs. PPV. de AF=20 %; GATK, Platypus y Lofreq superpuestos en la coordenada (1,1) . . . . .	38
4.3. Diagrama de dispersión de sensibilidad vs. PPV. de AF=5 %; GATK, Platypus y Lofreq superpuestos en la coordenada (1,1) . . . . .	39
4.4. Diagrama de dispersión de sensibilidad vs. PPV. de AF=1 %; FreeBayes y VarScan superpuestos . . . . .	40
4.5. Diagrama de dispersión de sensibilidad vs. PPV. de AF=0,05 % . . . . .	41
4.6. Sensibilidad, PPV y F1 score vs. AF . . . . .	43
4.7. Gráficos de dispersión de Sensibilidad vs. PPV. . . . .	45
4.8. Sensibilidad, PPV y F1 score( $\times 10^3$ ) vs. AF . . . . .	46
4.9. Gráficos de dispersión de Sensibilidad vs. PPV. . . . .	49
4.10. Sensibilidad, PPV y F1 score( $\times 10^{-3}$ ) vs. AF . . . . .	51

# Índice de cuadros

3.1. Listado completo de variantes asociadas a resistencia encontradas en el VHC tipo 1b, según Chen et al. 2016 . . . . .	31
--	----