Trabajo Práctico Cuatrimestral

Bioinformática

Grupo 4

Profesores:	Mariano Nicolás Breglia	
	Adrián Federico Pérez	
Alumnos:	Mia Arrigoni	61271
	Mila Langone	61273
	Julian Bergman	61819
	Brisa Rojas Silva	60535

Buenos Aires, Argentina, Mayo 2024

Contenidos

1. Marco teórico: Psoriasis

La psoriasis es una enfermedad inflamatoria crónica sistémica, de naturaleza inmunológica que afecta a millones de personas de todas las edades en todo el mundo. Se caracteriza por la presencia de lesiones cutáneas inflamadas, enrojecidas y cubiertas de escamas que pueden ser dolorosas y causar picor. Pueden presentarse en diversas áreas del cuerpo, como el cuero cabelludo, los codos, las rodillas, las manos y los pies. Esta enfermedad tiene un impacto significativo en la calidad de vida de quienes la padecen y afecta tanto su salud física como emocional.

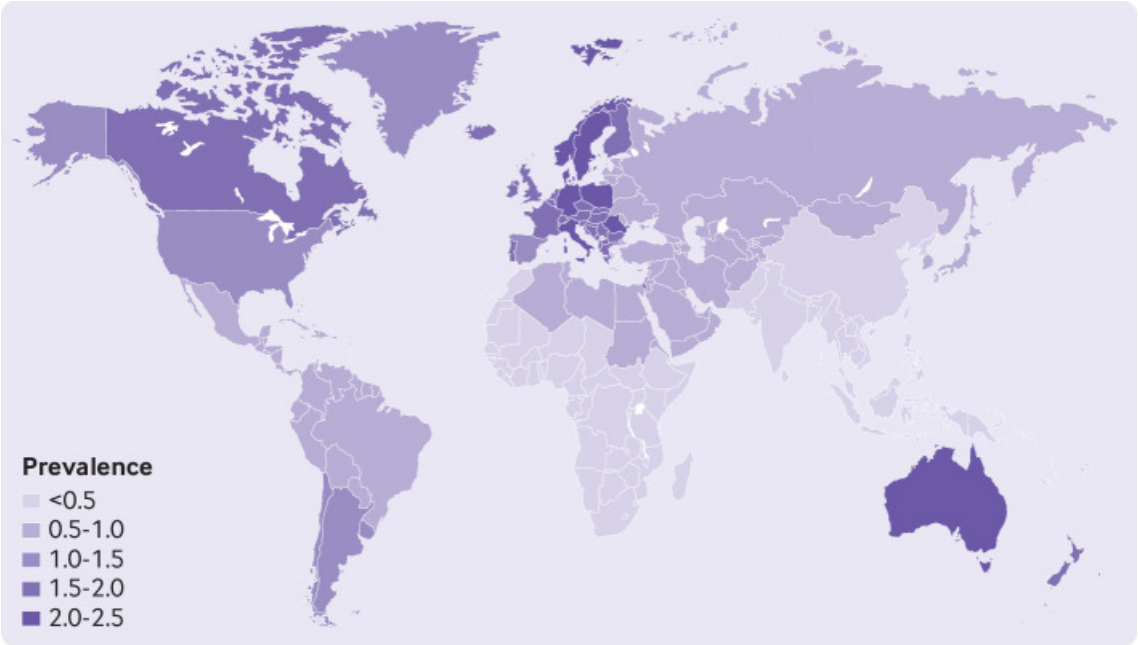

Figura 1.1: Prevalencia de por vida (diagnosticada por médico o dermatólogo) de psoriasis en adultos por país[1].

La causa exacta desencadenante de la psoriasis aún no se comprende completamente, pero se cree que es el resultado de una interacción compleja entre factores genéticos, inmunológicos y ambientales[2]. Se cree que algunos factores desencadenantes pueden ser infecciones, quemaduras por el sol, estrés, lastimaduras de la piel y algunos medicamentos.

Varios estudios han identificado posibles loci de susceptibilidad genética asociados con la psoriasis. Entre ellos se encuentran aquellos ubicados en los cromosomas 4q, 6p y 17q[2] [3] [4] [5] [6] y se nombran como PSORS1, PSORS2 o CARD14, PSORS3, PSORS4, PSORS5, PSORS6 y PSORS7[2] [3] [4] [5].

Se han realizado estudios de asociación para identificar posibles genes candidatos, así como análisis de expresión génica y estudios funcionales para comprender mejor los mecanismos subyacentes de la enfermedad. Este análisis es fundamental cuando se sospecha que la enfermedad puede estar relacionada con varios loci genéticos y alelos, como se ha observado en la psoriasis.

La evidencia sugiere que el locus 17q o 17q25 es uno de los principales involucrados en la psoriasis, como se ha demostrado en múltiples estudios[2;4].

Aunque se han identificado varios genes candidatos asociados con la psoriasis, este trabajo se centra en el locus 17q o 17q25 debido a su repetida implicación en múltiples estudios. En particular se va a hacer hincapié en el gen *Homo sapiens caspase recruitment domain family member 14* (*Homo sapiens CARD14*)[7].

El gen CARD14 codifica una proteína que juega un papel importante en la señalización y activación de la vía NF-κB, la cual tiene relación directa con la respuesta inmune del ser humano. La proteína asociada al gen abunda en la piel, regulando la supervivencia de las células relativas a este tejido. Se asocia con la psoriasis porque activa la vía NF-κB, aumentando la producción de citoquinas proinflamatorias que generan las manifestaciones clínicas de la enfermedad[8].

2. Metodología de trabajo: Kanban

El método Kanban es una práctica de metodologías ágiles para gestionar trabajo orientado a procesos con énfasis en la entrega puntual, sin sobrecarga para los miembros del equipo[9]. Permite aclarar prioridades y proporciona una forma de descubrir problemas en el flujo de trabajo y el proceso para que puedan resolverse.

El equipo de trabajo consta con miembros trabajando de manera remota y en diferentes horarios, lo cual representa un motivo clave para utilizar esta metodología. Cada miembro tiene sus propias responsabilidades y cronogramas, lo que hace que la flexibilidad y la claridad en la asignación de tareas sean indispensables. El sistema de extracción de trabajo, donde las tareas se adoptan por cada miembro conforme a su capacidad disponible, es otro aspecto crucial. De esta forma, se mejora el rendimiento global al eliminar la sobrecarga y asegurar plazos de completado más predecibles.

Kanban se basa en 5 principios básicos: visualizar, limitar el trabajo en curso, gestionar el flujo de trabajo, explicitar políticas de proceso y mejorar colaborativamente a un ritmo manejable y cómodo. Estos son pilares que garantizan la optimización continua y un trabajo en equipo eficiente. Además, se ajusta perfectamente a pro-

yectos con hitos o entregas bien definidas, permitiendo una planificación detallada y precisa.

A través de *Notion*[10] se diseñó una pizarra de trabajo que cuenta con tres columnas: no empezado (*Not Started*), en curso (*In Progress*) o terminado (*Done*). En cada columna, se posiciona cada tarea con una breve descripción y el responsable de la misma. Al utilizar una pizarra de trabajo se garantiza que se contabilicen todas las tareas, se muestre claramente quién es responsable de realizarlas y cuál es el estado de las mismas. En la Figura 2.1 se puede observar el tablero Kanban utilizado en uno de los avances del proyecto.

Avance #2

Aa Name	Assign	№ ID	Status
Interpretación del Alineamiento múltiple (TP1)	Mila	A2-19	Not started
Ej 7: Armar PPT	MIA ARRIGONI	A2-15	Not started
Ej 6G: Entrar en la base de datos de variantes genéticas dbSNP e intentar inter	JULIAN BERGMAN	A2-14	Not started
Ej 6F: Discutan brevemente en qué estructura o vías metabólicas específicas (	Brisa Rojas	A2-13	Not started
Ej 6E: Expliquen brevemente de qué componente celular forma parte su prote	MIA ARRIGONI	A2-12	In progress
Ej4: Bájense los motivos de las bases de datos PROSITE (archivo prosite.dat) y	JULIAN BERGMAN	A2-7	In progress
Ej4: Con EMBOSS, realice algún análisis sobre la secuencia de nucleótidos fast	Brisa Rojas JULIAN BER	A2-6	In progress
Ej 5: Crear un script que lleve a cabo dicho diseño de forma que sea parameti	Mila	A2-17	Done
Ej 5: A partir del transcipto seleccionado para el análisis de su investigación, c	Mila	A2-16	Done
Ej 6D: ¿Con cuántas otras proteínas interactúa el producto génico de su gen?	MIA ARRIGONI	A2-11	Done
Ej 6C: ¿Cuántos transcriptos y cuántas formas alternativas de splicing son cor	MIA ARRIGONI	A2-10	Done
Ej 6B: ¿Cuántos genes / proteínas homólogas se conocen en otros organismo	MIA ARRIGONI	A2-9	Done
Ej 6A: A partir del gen o proteína de interés para ustedes dar su link a NCBI-G	MIA ARRIGONI	A2-8	Done
Justificar "elección" de python + bash y demás tecnologías	Brisa Rojas	A2-4	Done
Justificación metodologías de trabajo	Mila	A2-3	Done

Figura 2.1: Tablero Kanban utilizado para el avance 2

3. Entorno de trabajo

Para el presente trabajo se decidió utilizar el sistema operativo Linux ya que proporciona un entorno estable y seguro, permitiendo un control preciso sobre el mismo y una amplia compatibilidad con herramientas bioinformáticas como BLAST y EMBOSS. Linux permite programar en bash, ofreciendo una facilidad notable para coordinar y gestionar el funcionamiento de los *scripts* de Python que llevan a cabo diversas tareas y análisis. La organización de esta manera facilita la automatización y la gestión de flujos de trabajo complejos en bioinformática.

Python es un lenguaje versátil y familiar para los miembros del equipo, lo que facilitó significativamente su incorporación en el entorno de trabajo. Su sintaxis clara

y legible permitió un desarrollo ágil de *scripts* para el análisis de datos biológicos utilizando la librería BioPython como principal herramienta.

Con el objetivo de gestionar las versiones del código de manera eficiente y ordenada, se creó un repositorio de GitHub donde se almacenaron los códigos y requerimientos para establecer el entorno en cualquier sistema y poder correr el programa de manera local.

3.1. *Pipeline* de trabajo

Para la automatización de los *steps* del presente trabajo se decidió incluir un *script* de *bash* que permita ejecutar los diferentes *scripts* de Python correspondientes a cada ejercicio. Esta decisión está orientada a desarrollar un trabajo sin *hardcode*, donde el archivo de entrada pueda ser modificado de manera simple y se pueda ejecutar el mismo *pipeline* sin necesidad de cambiar variables globales o locales.

La ventaja que trae el *pipeline* desarrollado es que permite alimentar el siguiente *step* con el *output* de un *step* anterior de manera automática, 'loggeando' paso a paso los errores y *outputs* que puedan ocurrir al ejecutar el programa.

Se incluyó además un mensaje en consola de cada *step*, para que el usuario pueda observar en qué estado está el código y en qué paso del *pipeline* está en cada momento, sin visualizar los *outputs*.

Además, se incluye un archivo de tipo .log que guarda los errores y un mensaje que indica el éxito de la ejecución con el marcador del día y horario en que fue ejecutado.

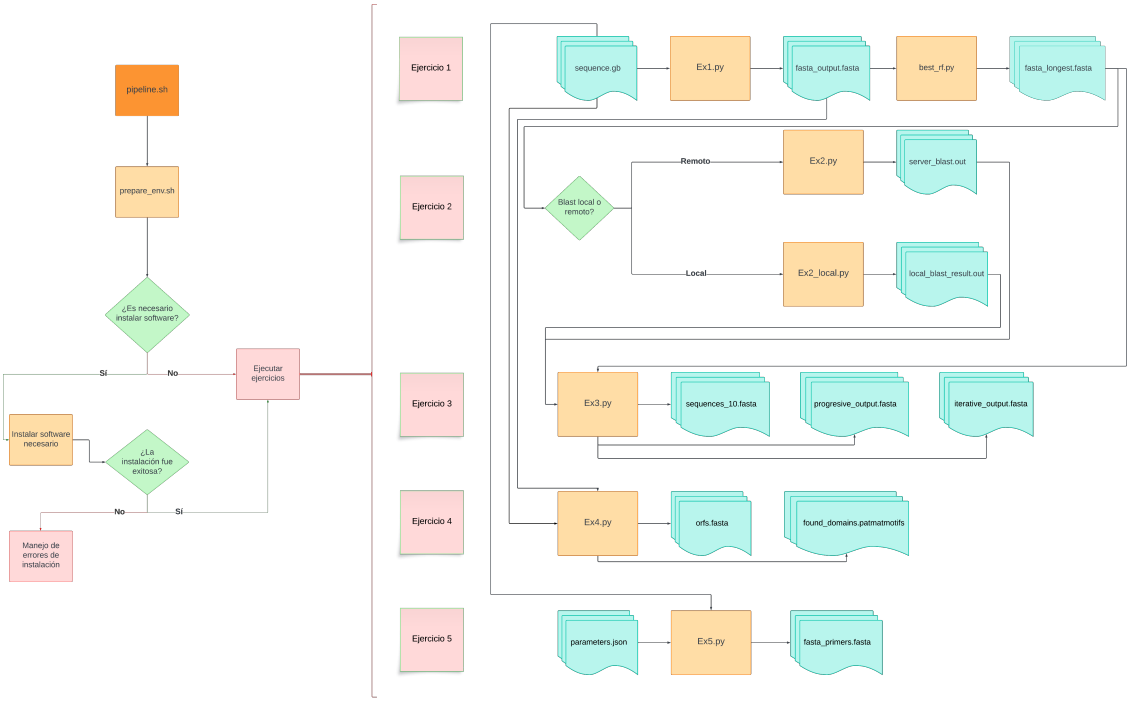

Figura 3.1: *Pipeline* de trabajo utilizado. Se pueden observar las decisiones tomadas en cada paso y qué archivos funcionan como *input* y *output* de cada *script*.

3.2. Preparación del entorno

Dado que para ejecutar el código fuente del presente trabajo es necesario tener instalados algunos programas y librerías, como paso inicial del *pipeline* se decidió incluir el *script* de *bash* (`prepare_env.sh`) que revisa si los mismos están instalados y actualizados. De ser así, se procede con la ejecución de los archivos en Python y si no, se instalan. En caso de que haya algún error con las instalaciones, el código no prosigue y se le avisa al usuario del error encontrado.

3.3. *Utils*

A lo largo del trabajo, es necesario tratar con distintos tipos de archivos (GenBank, FASTA, XML, JSON). Por este motivo, se decidió incluir un *script* auxiliar con funciones útiles para su respectiva correcta lectura y escritura. De esta forma, las mismas funciones pueden ser importadas desde distintos *scripts*, evitando código redundante.

De la librería `Seq.IO` de Biopython, se utilizan los atributos `parse` y `write` para lectura y escritura respectivamente.

4. Desarrollo

4.1. Ejercicio 1 - Procesamiento de secuencias

El archivo de secuencias GenBank *input* es obtenido del sitio web de NCBI[11] al realizar una consulta en la base de datos curada *Nucleotide* sobre el gen *Homo sapiens caspase recruitment domain family member 14 (Homo sapiens CARD14)*, filtrando por mRNA. El archivo para el presente análisis consiste de 4 transcriptos distintos del mismo. En particular, se eligieron las siguientes variantes:

# Transcripto	RefSeq ID	Longitud [bp]
1	NM_024110	4186
2	NM_052819	1633
3	NM_001257970	2612
5	NM_001366385	4717

Cuadro 4.1: Transcriptos elegidos del gen CARD14

Es importante aclarar que se utilizó un mismo archivo GenBank con 4 variantes distintas pero el código es lo suficientemente versátil y contempla los casos en donde el archivo pueda contener 1 sola secuencia o muchas más que cuatro, así como el error donde el archivo ingresado no sea válido.

El archivo *input* de secuencias es pasado como argumento `'-s'` al *script* de Python desde el *pipeline*.

Con la función `read_genbank_file`, se leen las secuencias de cada uno de los transcriptos y se genera su objeto `'Seq'` correspondiente.

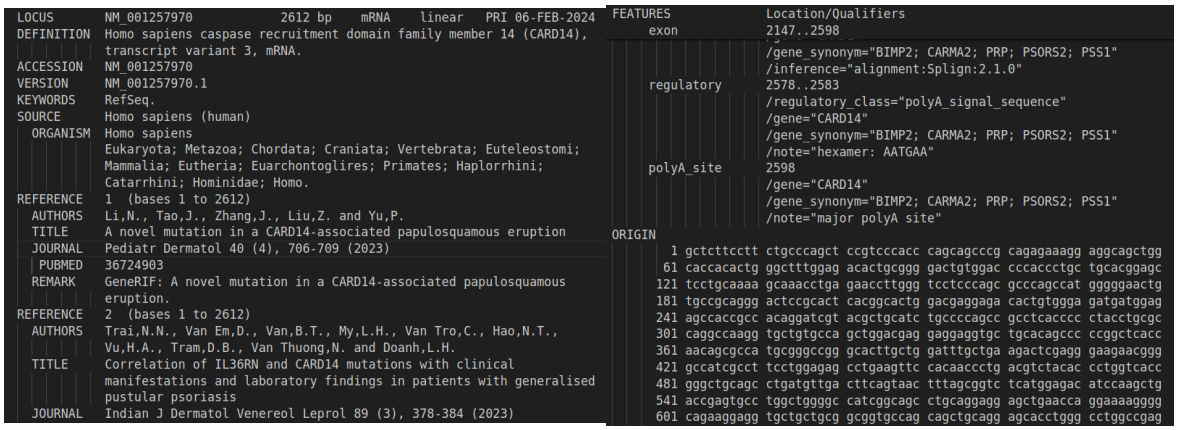

Figura 4.1: Archivo Genbank *input* de secuencias de nucleótidos

Previo a traducir las secuencias, es necesario tener en cuenta que una secuencia de nucleótidos tiene seis posibles marcos de lectura debido a la doble hebra del ADN. Cada hebra puede leerse de 5' a 3' y de 3' a 5'. Para cada hebra, la secuencia puede iniciarse desde tres posiciones diferentes, dado que según el código genético, un codón se compone de tres nucleótidos: comenzando desde el primer, segundo o tercer nucleótido. Luego, si la lectura comenzara desde el cuarto nucleótido, sería equivalente a comenzar desde el primero. Esto da lugar a un total de seis posibles marcos de lectura: tres en la hebra directa y tres en la hebra complementaria.

Como primer paso, se define una función para encontrar la posición de los codones de inicio en la secuencia. No todos los codones pueden iniciar la transcripción, el codón de inicio por defecto es 'ATG'. Para ello, se recorre la sencuencia de a 3 nucleótidos y se compara con 'ATG', si hay coincidencia perfecta se guarda la posición del primer nucleótido del codón.

Una vez obtenidos los posibles codones de inicio, se procede a traducir la secuencia de nucleótidos a aminoácidos. Para esto, se utiliza el atributo `'translate'` del objeto `'Seq'` de la librería BioPython[12].

Este atributo tiene como parámetros *default*:

```
translate(self, table='Standard', stop_symbol='*',
    to_stop=False, cds=False, gap=None)
```

En particular en este caso, se le pasa como parámetro `to_stop=True`, lo que permite que la traducción termine en el primer codón de STOP que se encuentre. El resto de los parámetros no se modifican, por lo que ese utilizará la tabla *Standard* de código genético y no se espera la presencia de GAPS.

```
>NM_001257970.10 <unknown description>
MGELCRRDSALTALDEETLWEMMESHRHRIVRCICPSRLTPYLRQAKVLCQLDEEEVLHS
PRLTNSAMRAGHLLDLLKTRGKNGAIAFLESLKFHNPDVYTLVTGLQPDVDFSNFSGLME
TSKLTECLAGAIGSLQEELNQEKGQKEVLLRRCQQLQEHLGLAETRAEGLHQLEADHSRM
KREVSAHFHEVLRLKDEMLSLSLHYSNALQEKELAASRCRSLQEELYLLKQELQRANMVS
SCELELQEQSLRTASDQESGDEELNRLKEENEKLRSLTFSLAEKDILEQSLDEARGSRQE
LVERIHSLRERAVAAERQREQYWEEKEQTLLQFQKSKMACQLYREKVNALQAQVCELQKE
RDQAYSARDSAQREISQSLVEKDSLRRQVFELTDQVCELRTQLRQLQAEPPGVLKQEART
REPCPREKQRLVRMHAICPRDDSDCSLVSSTESQLLSDLSATSSRELVDSFRSSSPAPPS
QQSLYKRVAEDFGEEPWSFSSCLEIPEGDPGALPGAKAGDPHLDYELLDTADLPQLESSL
QPVSPGRLDVSESGVLMRRRPARRILSQVTMLAFQGDALLEQISVIGGNLTGIFIHRVTP
GSAADQMALRPGTQIVMVDYEASEPLFKAVLEDTTLEEAVGLLRRVDGFCCLSVKVNTDG
YKRLLQDLEAKVATSGDSFYIRVNLAMEGRAKGELQVHCNEVLHVTDTMFQGCGCWHAHR
VNSYTMKDTAAHGTIPNYSR
>NM_001257970.11 <unknown description>
MLTSVTLAVSWRHPS
>NM_001257970.12 <unknown description>
MLFLCDGCFIYCK
>-NM_001257970.10 <unknown description>
MNISFAIYKASITKKQHLQQ
>-NM_001257970.11 <unknown description>
MSPPRSPLWPPGPGVASYNRPC
>-NM_001257970.12 <unknown description>
MPAAAALEHGVGDVQDLVAMHLQLPFGPALHGQVDPDVE
>NM_001366385.10 <unknown description>
```

Figura 4.2: Archivo FASTA *output* de secuencias aminoacídicas de cada transcripto y cada marco de lectura

En primer lugar, se obtienen las secuencias de aminoácidos correspondientes a los tres marcos de lectura de la hebra directa. Luego, se obtiene la hebra complementaria haciendo uso del atributo `'reverse_complement'`, de la librería BioPython. Por último, se ejecuta nuevamente la función de traducción para obtener las secuencias aminoacídicas correspondientes a los tres marcos de lectura de la hebra complementaria.

Finalmente, se guardan las secuencias aminoacídicas de cada transcripto de cada marco de lectura con su respectivo identificador *RefSeq* en un único archivo de formato FASTA (`fasta_output.fasta`) utilizando la función `write_fasta_file`.

4.2. Ejercicio 2 - BLAST

Para este ejercicio se realizó tanto la implementación de manera local conectandose al servidor público previsto por NCBI.

Como paso previo a la búsqueda en BLAST, se decidió buscar el mejor marco de lectura o ORF (*Open Reading Frame*) y realizar la búsqueda solo con esa secuencia. Para ello, se utilzó un *script* auxiliar (`best_rf.py`) que recibe el archivo FASTA de salida del ejercicio anterior y devuelve solamente el de mayor longitud que se corresponde con el primer ORF del transcripto 3. Se decidió considerar el parámetro de mayor longitud ya que se puede asumir que esta proteína será la de la naturaleza.

```
>NM_024110.40 <unknown description>
MGELCRRDSALTALDEETLWEMMESHRHRIVRCICPSRLTPYLRQAKVLCQLDEEEVLHS
PRLTNSAMRAGHLLDLLKTRGKNGAIAFLESLKFHNPDVYTLVTGLQPDVDFSNFSGLME
TSKLTECLAGAIGSLQEELNQEKGQKEVLLRRCQQLQEHLGLAETRAEGLHQLEADHSRM
KREVSAHFHEVLRLKDEMLSLSLHYSNALQEKELAASRCRSLQEELYLLKQELQRANMVS
SCELELQEQSLRTASDQESGDEELNRLKEENEKLRSLTFSLAEKDILEQSLDEARGSRQE
LVERIHSLRERAVAAERQREQYWEEKEQTLLQFQKSKMACQLYREKVNALQAQVCELQKE
RDQAYSARDSAQREISQSLVEKDSLRRQVFELTDQVCELRTQLRQLQAEPPGVLKQEART
REPCPREKQRLVRMHAICPRDDSDCSLVSSTESQLLSDLSATSSRELVDSFRSSSPAPPS
QQSLYKRVAEDFGEEPWSFSSCLEIPEGDPGALPGAKAGDPHLDYELLDTADLPQLESSL
QPVSPGRLDVSESGVLMRRRPARRILSQVTMLAFQGDALLEQISVIGGNLTGIFIHRVTP
GSAADQMALRPGTQIVMVDYEASEPLFKAVLEDTTLEEAVGLLRRVDGFCCLSVKVNTDG
YKRLLQDLEAKVATSGDSFYIRVNLAMEGRAKGELQVHCNEVLHVTDTMFQGCGCWHAHR
VNSYTMKDTAAHGTIPNYSRAQQQLIALIQDMTQQCTVTRKPSSGGPQKLVRIVSMDKAK
ASPLRLSFDRGQLDPSRMEGSSTCFWAESCLTLVPYTLVRPHRPARPRPVLLVPRAVGKI
LSEKLCLLQGFKKCLAEYLSQEEYEAWSQRGDIIQEGEVSGGRCWVTRHAVESLMEKNTH
ALLDVQLDSVCTLHRMDIFPIVIHVSVNEKMAKKLKKGLQRLGTSEEQLLEAARQEEGDL
DRAPCLYSSLAPDGWSDLDGLLSCVRQAIADEQKKVVWTEQSPR
```

Figura 4.3: Archivo FASTA de secuencia aminoacídica correspondiente al ORF más largo

4.2.1. Implementación Local

Haciendo uso de la librería `subprocess`, se implementó una función que a partir de los parámetros ingresados, realice la búsqueda BLAST. La función tiene como *input* el archivo de secuencias aminoacídicas de cada transcripto de tipo FASTA y como *output* un archivo .out con los resultados de la búsqueda.

Los parámetros utilizados son:

```
blast_command = [
    "blastp",
    "-query", sequence,
    "-db", "swissprot",
    "-out", output,
    "-outfmt", "5"
    ]
```

De esta forma se realiza una búsqueda de proteínas, en la base de datos SwissProt y el archivo de salida será de tipo `.out`.

4.2.2. Implementación con servicio WEB

Se realizó también una implementación de BLAST de manera remota, accediendo a la API de BLAST en el servidor de NCBI. Esto lo hace a través del servicio cloud que provee el módulo NCBIWWW del paquete Blast de la librería Biopython. La función utilizada en este caso es `qblast()`.

El beneficio que trae esta alternativa es la posibilidad de ejecutar un BLAST a través de un archivo de código sin la necesidad de descargar una base de datos

extensa (como lo son las de proteínas). Además, se realiza una conexión segura hacia la API, devolviendo resultados certeros.

Específicamente, está desarrollada en el archivo `Ex2.py` la implementación al llamar a la función `qblast()`. En esta se ingresaron los parámetros *blastp, "swiss-prot", sequence y hitlist_size*. Se corre *blastp* porque se debe realizar un BLAST de proteína, que accede a la base de *SwissProt* (reconocida por estar curada). Se pide un *hitlist_size* de 20 por una cuestión de velocidad de ejecución, ya que una búsqueda mayor tarda mucho tiempo.

Una vez que se realiza el BLAST, los resultados se guardan en un archivo de salida que será de tipo `.out`.

4.2.3. Interpretación de los resultados BLAST

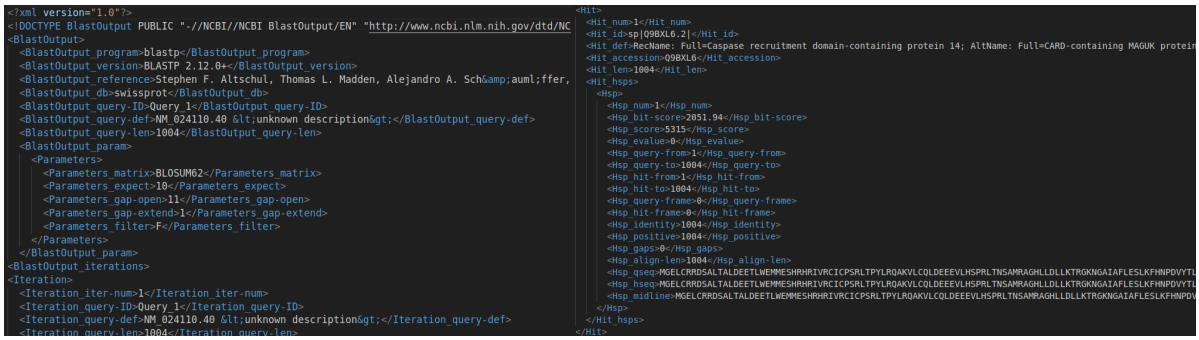

Figura 4.4: Archivo XML con resultados de la búsqueda en BLAST local

Los resultados del BLAST incluyen una lista de secuencias encontradas que tienen similitud con la secuencia de consulta, junto con estadísticas asociadas. Dentro de las estadísticas a las que se hace referencia están: el *query cover*, que es el porcentaje de la secuencia del *query* que cubre la alineación con la referencia; el *E-value*, que es una medida estadística que estima la probabilidad de obtener una coincidencia de alineación entre la secuencia de consulta y una secuencia en la base de datos simplemente por azar; y el *max value* que indica la fuerza de la mejor coincidencia encontrada. Con el resultado en BLAST, se encontraron 12 secuencias con similitudes a la de consulta. Podemos destacar que todas ellas tienen un *E-value* que oscila entre 0 y 2×10^{-7} , indicando que la coincidencia en la alineación no fue azarosa. En cuanto al *query cover*, vemos que hay 2 secuencias que devuelven un 100 %. La primera de ellas, además, tiene un porcentaje de identidad también del 100 %, ya que es la secuencia de consulta original. Analizando entonces la segunda

secuencia del BLAST, que posee un casi 77 % de porcentaje de identidad respecto de la secuencia original, junto con un bajo *E-value*, y altos valores de *max* y *total score*, podemos decir que nuestro *script* encontró correctamente el marco de lectura que codifica una proteína en la naturaleza.

4.3. Ejercicio 3 - MSA

Para este ejercicio, en primer lugar, se define una función que limita los resultados de BLAST a los primeros 10 *hits*. Recibe como parámetros el archivo .out con los resultados de la búsqueda BLAST y devuelve un archivo .fasta reducido con las secuencias correspondientes a los *hits*.

A continuación, se definen dos funciones para realizar los alineamientos múltiples.

En primer lugar, `progresive_alignment()`, realiza un alineamiento progresivo. Recibe como parámetros un archivo de las secuencias *input* en formato fasta y el nombre del archivo *output*. Para realizar el alineamiento se hace uso de la herramienta `subprocess` que permite llamar a la herramienta Clustalo.

```
>sp|Q9BWT7.2| <unknown description>
---------------DALWERIEGVRHRLARALNPAKLTPYLRQCRVIDEQDEEEVLST
YRFPCRVNRTGRLMDILRCRGKRGYEAFLEALEFYYPEHFTLLTGQEPAQRCSMILDEEG
PEGLTQFLMTEVRRLREARKSQLQREQQLQARGRVLEEE----------RAGLEQRLRDQ
QQAQERCQRLREDWEAGSLELLRLKDENYMIAMRLAQLSEEKNSAVLRSRDLQLAVDQLK
LKVSR--LEEECAL------LRRARGPPPGAEEKEKEKEKEKEPDNVDLVSELRAENQRL
TASLRELQEGLQQEASRPGAPGSERILLDILEHDWREAQDSRQELCQKLHAVQGELQWAE
ELRDQYLQEMEDLRLKHRTLQKDCDLYKHRMATVLAQLEEIEKERDQAIQSRDRIQLQYS
QSLIEKDQYRKQVRGLEAERDELLTTLTSLEGTKALLEVQLQRAQGGTCLKAC-------
---ASSHSLCSNLSSTWSLSEFPSPLGGPEATGEAAV----------------------
---------------------------------------------MGGPEPHNSEE--ATD
SEKEINRLSILPFPPSAGSILRRQREEDPAP--PKRSFSSMSDI----------------
--------TGSV-----TLKPWSPGLSSSSSSDSVWPLGKPEGLLARGCGLDFLNRSLAI
RVSGRSPPGGPEPQDKGPDGL--------------------------SFYGDRWSGAVV
RRVLSGPGSARM--EPREQRVEA----------AGLEGACLEAEAQQRTLLWNQGSTLP
SLMDSKACQ-------SFHEALEAWAK---------GPGAEPFYIRANLTLPERADPHAL
CVKAQEILRLVDSAYKRR-QEWFCTRVDPLTLRDL-DRGTVPNYQRAQQLL-----EVQE
KCL----PS--------SRH--------------------RGPRSNLKKRALDQLRLVR
PKPVG--------AP----AGDSPD-Q-----LLLEPC----------AE--PERSLRP
YSLVRPLLVSALRPVVLLPECLAPRLIRNLLDLPSSRL-DFQVCPAESLSGEELCPSSAP
G-AP-KAQPATPGLGSRI-RAIQESVGKK--HCLLELGARGVRELVQNEIYPIVIHVEVT
EKNVRE----VRGLLGRPGWRDSELLRQCRGSEQVLWGLPCSW-VQVPAHEWGHAEELAK
VVRGRILQEQARLVWVE----
>sp|Q9EPY0.1| <unknown description>
------------------WSALESFRVKLISVIDPSRITPYLRQCKVLNPDDEEQVLSD
```

Figura 4.5: Archivo FASTA *output* de alineamiento múltiple progresivo

Para el alineamiento iterativo, se define otra función (`iterative_alignment`) que recibe los mismos parámetros. En este caso, se hace uso de la herramienta Muscle con `subprocess` nuevamente.

Figura 4.6: Archivo FASTA *output* de alineamiento múltiple iterativo

Por último, para evaluar la *performance* de ambos algoritmos se hizo uso de la librería `time` y se comparó el tiempo de ejecución de cada una de las funciones que se visualiza en un *DataFrame* en el archivo .log.

```
El tiempo de ejecucion de cada algoritmo es:
   Función de Alineamiento  Tiempo de Ejecución (s)
0              Progresivo                 1.962730
1               Iterativo                 1.352676
```

Figura 4.7: Tiempo de ejecución de ambos algoritmos para comparación de *performance*

4.3.1. Interpretación

Las figuras 4.5 y 4.6 muestran el alineamiento múltiple del *ORF* más largo y una de las entradas del blast realizado en el ejercicio anterior. Para poder tener una visión más general de las secuencias alineadas se ingresaron los archivos salientes a SeaView, donde se puede ver la longitud de las secuencias alineadas, los matches, mismatches y gaps, junto con algunas estadísticas. Pueden verse en las figuras 4.8 y 4.9.

En el alineamiento progresivo, la longitud del MSA es de 1281 sitios, pero ninguno de ellos estaba completo. Esto sugiere que las secuencias podrían estar bastante divergentes en esos sitios específicos, sin regiones altamente conservadas entre las especies analizadas.

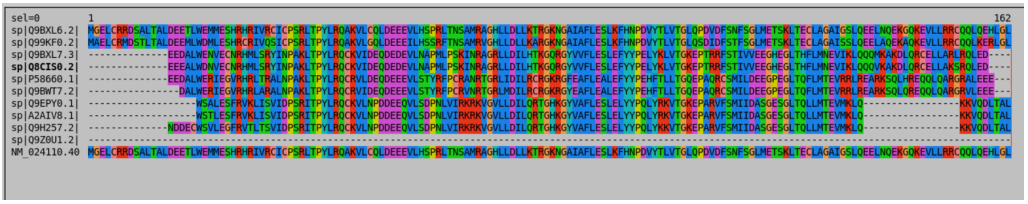

Figura 4.8: Primeras posiciones del Seaview del alineamiento progresivo

Con respecto al alineamiento iterativo, hay 1187 sitios de longitud, siendo algo más compacto y con menor cantidad de gaps. De estos 1187, 259 sitios estaban completos y conservados. En la imagen se pueden ver que hay columnas completas

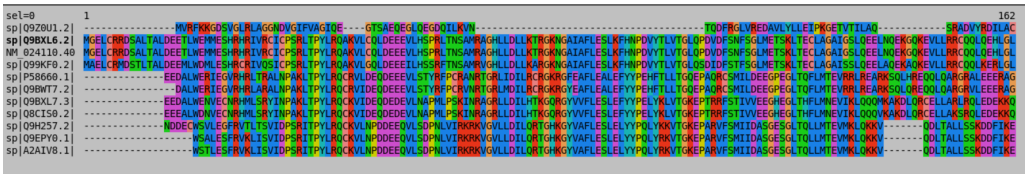

Figura 4.9: Primeras posiciones del Seaview del alineamiento iterativo

Según las estadísticas mostradas por SeaView, en el caso del alineamiento progresivo no hay sitios informativos, mientras que en el alineamiento iterativo hecho por MUSCLE se encuentra que el 80.3 % de los sitios completos son informativos (208/259). Esto permite inferir relaciones evolutivas o funcionales.

Finalmente, se puede analizar el tiempo de ejecución, que según los resultados obtenidos contradice lo que se esperaba. El tiempo es menor en el caso del alineamiento iterativo contra el progresivo. Se investigó sobre la librería MUSCLE y se concluyó que puede suceder esto porque es muy eficiente en su implementación por los algoritmos internos. Ya que optimizan su rendimiento en cada iteración en conjuntos de datos que no son grandes (como lo es este caso) [13].

4.4. Ejercicio 4 - EMBOSS

4.4.1. Implementación Local

Para la implementación local de la búsqueda de ORFs y la identificación de motivos proteicos utilizando EMBOSS, se siguieron los siguientes pasos:

1. **Instalación de EMBOSS:** Se verificó si EMBOSS estaba instalado en el sistema y, de no ser así, se procedió a su instalación utilizando los comandos adecuados del sistema operativo.

2. **Configuración de la base de datos PROSITE:** Se descargaron los archivos necesarios de la base de datos PROSITE y se almacenaron en la ubicación apropiada dentro del sistema. Posteriormente, se procesaron estos archivos utilizando la herramienta `prosextract` para prepararlos para su uso con EMBOSS.

3. **Identificación de ORFs:** Utilizando la función `getorf` de EMBOSS, se identificaron los marcos de lectura abiertos (ORFs) en la secuencia de nucleótidos proporcionada. El archivo de secuencia de entrada se procesó y se generó un archivo de salida `(orfs.fasta)` con las secuencias de los ORFs encontrados.

4. **Análisis de Motivos Proteicos:** Se utilizó la función `patmatmotifs` de EMBOSS para buscar motivos proteicos en las secuencias de aminoácidos. Esta función tomó como entrada las secuencias de proteínas y utilizó la base de datos PROSITE para identificar patrones de motivos, generando un archivo de salida (`found_domains.patmatmotifs`) con los resultados del análisis. Esta función devolvió un reporte detallado de los motivos encontrados en las secuencias analizadas. Por ejemplo, para la secuencia `NM_001257970.10`, se identificaron dos motivos:

   - **Motivo: LEUCINE_ZIPPER**
     - **Longitud:** 22 aminoácidos
     - **Inicio:** posición 385 de la secuencia
     - **Fin:** posición 406 de la secuencia
     - **Secuencia:** `LRRQVFELTDQVCELRTQLRQL`
     - **Descripción:** Comúnmente encontrado en factores de transcripción, facilita la dimerización de hélices alfa específicas, permitiendo la interacción de proteínas con el ADN.
   - **Motivo: TYR_PHOSPHO_SITE_2**
     - **Longitud:** 8 aminoácidos
     - **Inicio:** posición 220 de la secuencia
     - **Fin:** posición 227 de la secuencia
     - **Secuencia:** RSLQEELY
     - **Descripción:** Este motivo es un sitio de fosforilación de tirosina, crucial para la regulación de la actividad proteica mediante la adición de un grupo fosfato.

4.4.2. Implementación WEB

A modo de comparación, también se obtuvieron los ORFs utilizando una herramienta online de EMBOSS (disponible en https://www.bioinformatics.nl/cgi-bin/emboss/getorf).

La GUI pide una secuencia y permite ingresar ciertos parámetros como:

- Tamaño mínimo y máximo de nucleótidos a reportar.
- Código génetico a utilizar.
- Indicar si los codones de start deben ser cambiados a metionina.
- Formato del archivo de *output*.
- Otros parámetros.

Al usar el archivo *sequence.gb* se obtiene el *output* siguiente: https://www.bioinformatics.nl/emboss-explorer/output/813020/

Algunos los mantuvimos iguales, como el formato del archivo de salida, pero otros difieren. Algo a notar es la diferencia entre el número de ORFs encontrados por nosotros anteriormente y los provistos por esta herramienta modificando los valores de las opciones brindadas . Esto se verificó usando un comando de *bash* debido al formato que tiene el archivo FASTA usando

```
grep -c '^>' orfs.fasta
```

para contar entradas.

En particular, hay 2 parámetros que variamos para ver su influencia en los resultados obtenidos:

1. **Tamaño mínimo nucleótidos a reportar:** Al aumentar este valor, se filtran ORFs más cortos, reduciendo el número total de ORFs identificados. Por ejemplo, si se establece un tamaño mínimo de 30 nucleótidos, se excluirán todos los ORFs menores a esta longitud. Si se disminuye el tamaño mínimo, se detectarán más ORFs, incluyendo aquellos más cortos.

2. **Identificación de ORFs en la secuencia inversa:** Al buscar ORFs en ambas hebras de ADN (directa e inversa), se puede duplicar el número de ORFs identificados. Si esta opción está desactivada, solo se reportarán ORFs en la hebra directa.

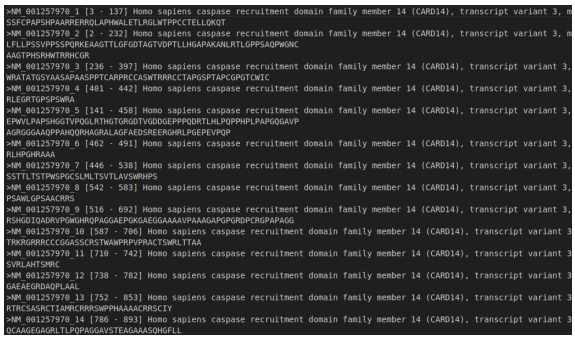

Figura 4.10: Archivo *output* de ORFs encontrados

4.5. Ejercicio 5 - Diseño de primers

Para el diseño de *primers* es necesario tener en cuenta los siguientes parámetros:

- **Tamaño en pares de base:** de 18 a 24
- **Contenido GC:** entre 50 % y 60 %
- **Evitar GC en los extremos**
- **Temperatura de *melting*:** menor a 67°

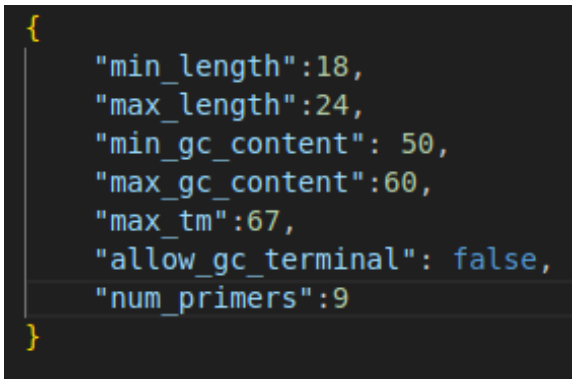

Figura 4.11: Archivo JSON con parámetros para el diseño de *primers*

Cada parámetro es validado por su función correspondiente que recupera los valores específicos para este análisis de un archivo de tipo JSON. Esto hace énfasis en el enfoque de automatización y parametrización que se mantiene a lo largo de todo el trabajo.

Se definen las siguientes funciones: `valid_gc_content`, `valid_length`, `valid_tm` y `has_gc_terminal`. Cada función es autónoma y concisa, aunque no irreducible.

Se consideró la relación de compromiso entre la claridad del código y la cantidad de líneas para facilitar la reproducibilidad por parte de futuros desarrolladores o evaluadores.

Además, se define otra función, `validate_primer`, que invoca las funciones mencionadas y verifica la condición de GC terminal. Esta función retorna un valor booleano indicando si el *primer* diseñado pasó la validación o no.

Una vez establecida la condición de validación, se generan la cantidad de *primers* válidos establecidos en el JSON tanto para la hebra directa como para la hebra complementaria. Es una buena práctica en bioinformática diseñar tanto los *primers* para para polimerizar la hebra *forward* como la reversa.

Las secuencias obtenidas para cada primer de cada transcripto se guardan en un archivo de tipo FASTA bajo un identificador único compuesto de su RefSeq, hebra forward (fwd) o complementaria (rev) y el número de *primer* (entre 0 y 4).

Nuevamente, tanto el archivo Genbank de secuencias *input* como el JSON con los parámetros son pasados como argumento desde el *pipeline*. Se hace uso de las funciones `read_genbank_file`, `read_json` y por último `write_fasta_file` para escribir el archivo *output* FASTA.

```
>NM_001257970.1_fwd_primer_0 <unknown description>
TCTTCCTTCTGCCCAGCT
>NM_001257970.1_fwd_primer_1 <unknown description>
TCTTCCTTCTGCCCAGCTCCGT
>NM_001257970.1_fwd_primer_2 <unknown description>
TTCCTTCTGCCCAGCTCCGT
>NM_001257970.1_fwd_primer_3 <unknown description>
AGCAGCCCGCAGAGAAAGGA
>NM_001257970.1_fwd_primer_4 <unknown description>
AGAGAAAGGAGGCAGCTGGCA
>NM_001257970.1_fwd_primer_5 <unknown description>
AGAGAAAGGAGGCAGCTGGCACCA
>NM_001257970.1_fwd_primer_6 <unknown description>
AGAAAGGAGGCAGCTGGCA
>NM_001257970.1_fwd_primer_7 <unknown description>
AGAAAGGAGGCAGCTGGCACCA
>NM_001257970.1_fwd_primer_8 <unknown description>
AGAAAGGAGGCAGCTGGCACCACA
>NM_001257970.1_rev_primer_0 <unknown description>
AGTGATCTTGGCAGGCTTCCCT
>NM_001257970.1_rev_primer_1 <unknown description>
AGTGATCTTGGCAGGCTTCCCT
>NM_001257970.1_rev_primer_2 <unknown description>
AGTGATCTTGGCAGGCTTCCCT
```

Figura 4.12: Archivo *output* de secuencias de los *primers* diseñados para cada transcripto y cada hebra

4.6. Ejercicio 6 - Trabajo con bases de datos biológicas

4.6.1. Ejercicio 6.1

La proteína codificada por el gen CARD14 (*caspase recruitment domain family member 14*) pertenece a la familia MAGUK (quinasa guanilato asociada a la membrana) y actúa como una proteína andamiaje. Esta proteína facilita la adhesión celular, la transducción de señales y el control de la polaridad celular. CARD14 interactúa específicamente con BCL10 (Fig. 4.1, una proteína reguladora positiva de la apoptosis y activación de NF-κB) y participa en la activación de NF-κB (Fig. 4.2, un factor de transcripción crucial para la respuesta inmune y la inflamación). El splicing alternativo del gen CARD14 resulta en múltiples variantes de transcripción, permitiendo la producción de diferentes isoformas de la proteína. Mutaciones en el gen CARD14 se han asociado con enfermedades inflamatorias de la piel, como la psoriasis en placas y la dermatitis atópica, debido a una activación anormal de NF-kappaB que conduce a la inflamación crónica[7].

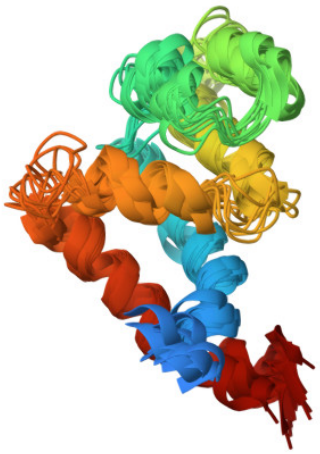

Figura 4.13: Unidad biológica de BCL10 (PDB: 2MB9)

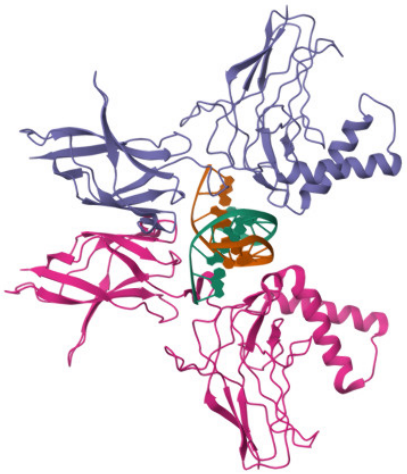

Figura 4.14: Unidad biológica de NF-kappaB (PDB: 1A3Q)

4.6.2. Ejercicio 6.2

Gene: CARD14 ENSG00000141527

Description	caspase recruitment domain family member 14 [Source:HGNC Symbol;Acc:HGNC:16446]
Gene Synonyms	BIMP2, CARMA2, PSORS2
Location	Chromosome 17: 80,169,992-80,216,073 forward strand. GRCh38:CM000679.2
About this gene	This gene has 30 transcripts (splice variants), 184 orthologues, 3 paralogues and is associated with 4 phenotypes.

Figura 4.15: Resultados *Ensembl*

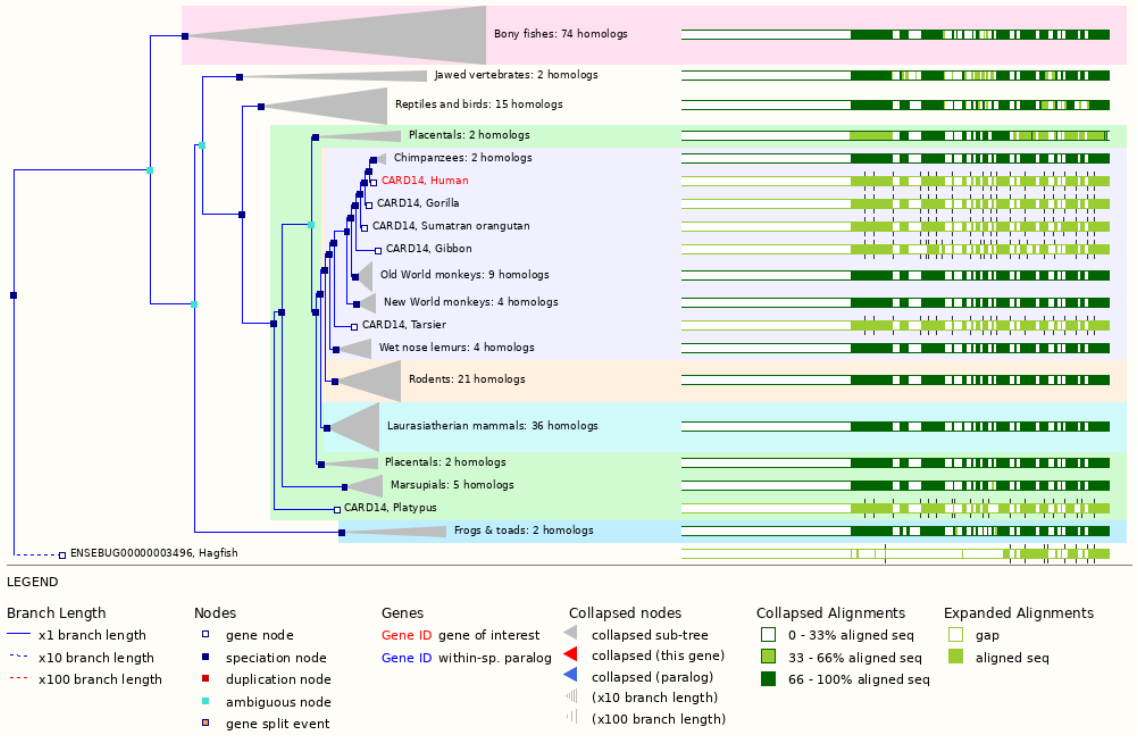

Figura 4.16: Resultados Ensembl en formato *gene tree.*

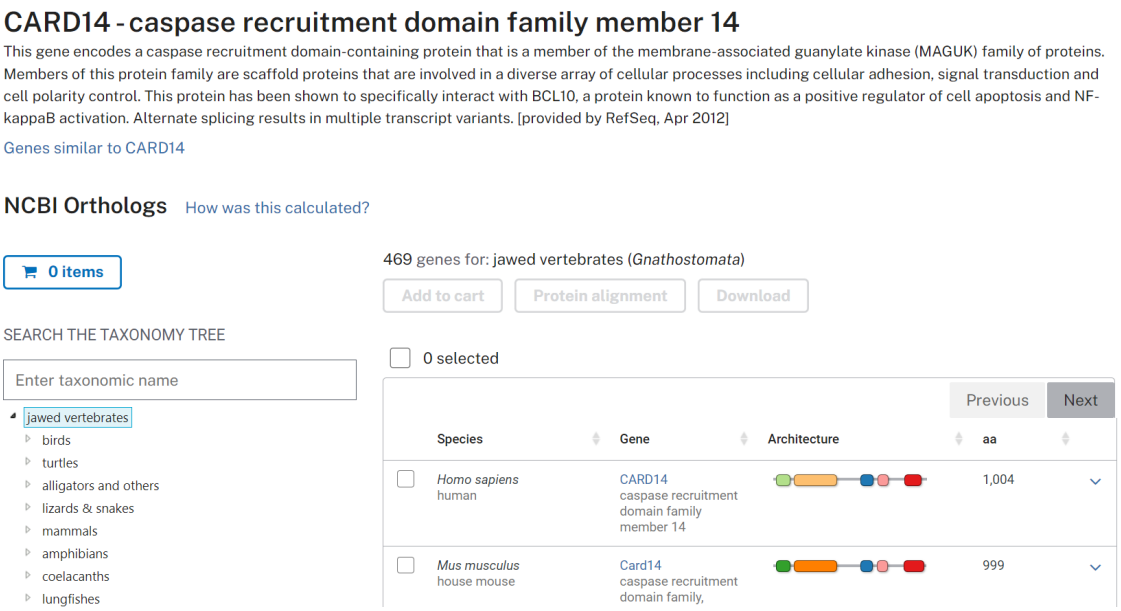

Figura 4.17: Resultados NCBI

Para la búsqueda de genes o proteínas homólogas, de la base de datos *Ensembl* se obtuvieron 184 genes ortólogos y 3 parálogos. De la base de datos de NCBI se obtienen 469 ortólogos. Esta útlima se enfoca específicamente en la identificación de genes homólogos, mientras que *Ensembl* abarca información más amplia sobre genomas completos. Además, la presentación de los resultados difiere entre ambas plataformas. Como se puede ver en la figura 4.17, *HomoloGene* puede presentarlos en forma de tabla, mientras que *Ensembl* puede además hacerlo en formato de *gene tree* (figura 4.16). Este último tiene un alcance mucho más amplio, ya que además de la identificación de genes homólogos, incluye datos detallados sobre la estructura genómica y anotaciones funcionales, permitiendo también visualizar la proximidad evolutiva entre los genes.

4.6.3. Ejericio 6.3

En cuanto a los transcriptos y las formas de *splicing* alternativo, de acuerdo a la base de datos de NIH (figura 4.18), el gen CARD14 produce múltiples transcriptos con variantes de *splicing* alternativo, pero 3 son de mayor interés para este análisis. La variante 1 codifica la isoforma 1, una proteína grande que contiene un dominio de reclutamiento de caspasa (CARD), esencial para la interacción con BCL10 y la activación de NF-kappaB, involucrada en la señalización celular y la respuesta inmune. La variante 2 codifica la isoforma 2, una proteína más pequeña que también

incluye el dominio CARD, pero con diferencias en otras regiones que pueden alterar su función y su interacción con otras proteínas. La variante 3 codifica la isoforma 3, una proteína que conserva el dominio CARD pero presenta diferencias adicionales en su secuencia que pueden afectar su localización subcelular y su capacidad de interactuar con diferentes moléculas de señalización, permitiendo así una variedad de funciones en distintos contextos celulares. Estas variaciones permiten que CARD14 desempeñe múltiples roles en la regulación de procesos celulares.

Los identificadores NM_XXXXXX en NCBI Gene corresponden a transcripciones que han sido validadas experimentalmente. Por otra parte, los identificadores XM_XXXXXX y XP_XXXXXX se refieren a transcripciones predichas de mRNA y proteinas respectivamente. Estos últimos se respaldan en predicciones basadas en análisis computacionales y comparaciones de secuencias.

Gene ID	Gene symbol	Transcript	Length (nt)	Protein	Length (aa)	Protein name	Isoform
79092	CARD14	NR_047566.2	4138				
79092	CARD14	XM_047436713.1	4474	XP_047292669.1	1004	caspase recruitment domain-containing protein ...	X1
79092	CARD14	XM_047436719.1	5328	XP_047292675.1	936	caspase recruitment domain-containing protein ...	X3
79092	CARD14	XM_047436714.1	4456	XP_047292670.1	1004	caspase recruitment domain-containing protein ...	X1
79092	CARD14	NM_001366385.1	4717	NP_001353314.1	1004	caspase recruitment domain-containing protein ...	1
79092	CARD14	XM_011525213.2	4394	XP_011523515.1	1004	caspase recruitment domain-containing protein ...	X1
79092	CARD14	XM_011525216.2	4196	XP_011523518.1	1004	caspase recruitment domain-containing protein ...	X1
79092	CARD14	XM_011525218.2	4231	XP_011523520.1	1004	caspase recruitment domain-containing protein ...	X1
79092	CARD14	XM_047436715.1	4492	XP_047292671.1	1004	caspase recruitment domain-containing protein ...	X1
79092	CARD14	XM_047436717.1	4672	XP_047292673.1	1004	caspase recruitment domain-containing protein ...	X1
79092	CARD14	XM_047436716.1	4376	XP_047292672.1	1004	caspase recruitment domain-containing protein ...	X1
79092	CARD14	NM_024110.4	4186	NP_077015.2	1004	caspase recruitment domain-containing protein ...	1
79092	CARD14	XM_047436718.1	4166	XP_047292674.1	1003	caspase recruitment domain-containing protein ...	X2
79092	CARD14	XM_047436720.1	3038	XP_047292676.1	854	caspase recruitment domain-containing protein ...	X4
79092	CARD14	XM_047436721.1	2796	XP_047292677.1	740	caspase recruitment domain-containing protein ...	X5
79092	CARD14	XM_047436722.1	2669	XP_047292678.1	678	caspase recruitment domain-containing protein ...	X6
79092	CARD14	NM_001257970.1	2612	NP_001244899.1	740	caspase recruitment domain-containing protein ...	3

Figura 4.18: Salida de NIH. Se muestran las primeras 17 salidas de un total de 47.

Los búsqueda en *Ensembl* (figura 4.19 muestra 30 trasncriptos alternativos. Como se puede observar, hay 7 transcriptos sumamente relevantes asociados a proteinas codificantes. Dentro de estos, podemos encontrar tamaños de proteínas muy variados, que van desde 70 hasta 1000 aminoácidos aproximadamente. Otros 9 tienen codones de *stop* prematuros y sufren un proceso de degradación; 6 no tienen región de codificación definida; y 8 todavía tienen intrones, por lo que no codifican para proteína.

Transcript ID	Name	bp	Protein	Biotype	CCDS	UniProt Match	RefSeq Match	Flags
ENST00000648509.2	CARD14-215	4717	1004aa	Protein coding	CCDS11768	Q9BXL6-1	NM_001366385.1	MANE Select Ensembl Canonical GENCODE basic APPRIS P1
ENST00000571427.2	CARD14-203	4732	1004aa	Protein coding	CCDS11768	Q9BXL6-1	-	GENCODE basic APPRIS P1 TSL:5
ENST00000573882.5	CARD14-209	4537	1004aa	Protein coding	CCDS11768	Q9BXL6-1	-	GENCODE basic APPRIS P1 TSL:5
ENST00000651672.1	CARD14-221	4475	1013aa	Protein coding		A0A494C1N2	-	GENCODE basic
ENST00000344227.6	CARD14-201	4147	1004aa	Protein coding	CCDS11768	Q9BXL6-1	-	GENCODE basic APPRIS P1 TSL:1
ENST00000570421.5	CARD14-202	2607	740aa	Protein coding	CCDS58605	Q9BXL6-2	-	GENCODE basic TSL:1
ENST00000575465.6	CARD14-211	582	70aa	Protein coding		I3L1Z7	-	TSL:5 CDS 3' incomplete
ENST00000703566.1	CARD14-223	4305	678aa	Nonsense mediated decay		A0A994J6G4	-	-
ENST00000703567.1	CARD14-224	4287	678aa	Nonsense mediated decay		A0A994J6G4	-	-
ENST00000650867.1	CARD14-218	4082	854aa	Nonsense mediated decay		A0A494C0J4	-	-
ENST00000651068.1	CARD14-219	3838	553aa	Nonsense mediated decay		A0A494BZY0	-	-
ENST00000575500.5	CARD14-212	3766	493aa	Nonsense mediated decay		I3L414	-	TSL:2
ENST00000703568.1	CARD14-225	3766	493aa	Nonsense mediated decay		I3L414	-	-
ENST00000651388.1	CARD14-220	2429	636aa	Nonsense mediated decay		A0A494C199	-	-
ENST00000571450.1	CARD14-204	1554	475aa	Nonsense mediated decay		I3L3F1	-	TSL:2 CDS 5' incomplete
ENST00000648128.1	CARD14-214	985	84aa	Nonsense mediated decay		A0A3B3ITQ9	-	CDS 5' incomplete
ENST00000703569.1	CARD14-226	3927	No protein	Protein coding CDS not defined		-	-	-
ENST00000703570.1	CARD14-227	3052	No protein	Protein coding CDS not defined		-	-	-
ENST00000573754.5	CARD14-208	1633	No protein	Protein coding CDS not defined		-	-	TSL:1
ENST00000703572.1	CARD14-229	1507	No protein	Protein coding CDS not defined		-	-	-
ENST00000573489.5	CARD14-207	555	No protein	Protein coding CDS not defined		-	-	TSL:4
ENST00000574148.5	CARD14-210	468	No protein	Protein coding CDS not defined		-	-	TSL:4
ENST00000652599.1	CARD14-222	3929	No protein	Retained intron		-	-	-
ENST00000650806.1	CARD14-217	3595	No protein	Retained intron		-	-	-
ENST00000703571.1	CARD14-228	2399	No protein	Retained intron		-	-	-
ENST00000649277.1	CARD14-216	2378	No protein	Retained intron		-	-	-
ENST00000575666.1	CARD14-213	2358	No protein	Retained intron		-	-	TSL:2
ENST00000703573.1	CARD14-230	2118	No protein	Retained intron		-	-	-
ENST00000571861.5	CARD14-205	1451	No protein	Retained intron		-	-	TSL:1
ENST00000572838.1	CARD14-206	553	No protein	Retained intron		-	-	TSL:4

Figura 4.19: Salida de Ensembl

Nuevamente se observa una diferencia en la cantidad de resultados que provee cada base de datos. Las diferencias en los resultados entre NCBI y Ensembl pueden atribuirse a varios factores. NCBI combina anotaciones manuales y automáticas, con transcriptos validados experimentalmente (NM_) y predicciones computacionales (XM_). Ensembl utiliza principalmente anotaciones automáticas basadas en modelos predictivos y análisis de secuencias, integrando datos de múltiples fuentes, lo que puede resultar en más transcriptos identificados.

4.6.4. Ejercicio 6.4

En NCBI se obtienen 17 interacciones (figura 4.20) con CARD14. Uniprot ofrece también una varidad de interacciones posibles (figura 4.21), pero se hará foco en las de tipo Proteína-Proteína, que ofrece 3 bases de datos distintas. Se optó por BiogGRID, que muestra 22 interacciones (figura 4.22).

De las interacciones que fueron devueltas por las bases de datos, se resaltan las siguientes:

- ***B-cell lymphoma/leukemia 10 (BCL10)***: Desempeña un papel crucial en la señalización inmune adaptativa e innata al conectar proteínas con dominio CARD con la activación inmune. Actúa canalizando esta señalización a través de CARD9, CARD11 y CARD14 para activar las vías NF-*kappa-B* y MAP quinasa p38 (MAPK11, MAPK12, MAPK13 y/o MAPK14), lo que estimula la expresión de genes proinflamatorios. Es reclutado por proteínas activadas con dominio CARD, formando un complejo que incluye a BCL10 y MALT1, promoviendo así la activación de estas vías y la producción de citocinas y quimiocinas proinflamatorias.

- ***Ubiquitin-associated domain-containing protein 1 (UBAC1)***: Componente no catalítico del complejo KPC, una ligasa de ubiquitina E3 que poliuquiquitina proteínas objetivo como CDKN1B y NFKB1. El complejo KPC regula la señalización de NF-*kappa-B* al madurar NFKB1 mediante la ubiquitinación de su precursor p105[14].

- ***Mucosa-associated lymphoid tissue lymphoma translocation protein 1 (MALT1)***: Proteasa que potencia la activación inducida por BCL10 al formar complejos CBM, canalizando la señalización inmune adaptativa e innata a través de proteínas CARD (CARD9, CARD11 y CARD14) para activar NF-*kappa-B* y la quinasa MAP p38, estimulando genes proinflamatorios[15].

Interactions

Products	Interactant	Other Gene	Complex	Source	Pubs	Description
NP_077015.1	NP_003912.1	BCL10		BIND	PubMed	CARD14 interacts with BCL10.
Q9BXL6	O95999	BCL10		HPRD	PubMed	
Q9BXL6	Q15645	TRIP13		HPRD	PubMed	
BioGRID:122540	BioGRID:114429	BCL10		BioGRID	PubMed	Affinity Capture-Western; Reconstituted Complex; Two-hybrid
BioGRID:122540	BioGRID:121581	HECW2		BioGRID	PubMed	Affinity Capture-MS
BioGRID:122540	BioGRID:609077	HULC		BioGRID	PubMed	Affinity Capture-MS
BioGRID:122540	BioGRID:116098	MALT1		BioGRID	PubMed	Affinity Capture-Western
BioGRID:122540	BioGRID:110378	MAP3K1		BioGRID	PubMed	Biochemical Activity
BioGRID:122540	BioGRID:124243	PCGF1		BioGRID	PubMed	Affinity Capture-MS
BioGRID:122540	BioGRID:111362	PLK1		BioGRID	PubMed	Affinity Capture-MS
BioGRID:122540	BioGRID:119524	PPME1		BioGRID	PubMed	Affinity Capture-MS
BioGRID:122540	BioGRID:117563	PRPF31		BioGRID	PubMed	Two-hybrid
BioGRID:122540	BioGRID:114977	RNF7		BioGRID	PubMed	Affinity Capture-Western; Two-hybrid
BioGRID:122540	BioGRID:114730	TRIP13		BioGRID	PubMed	Two-hybrid
BioGRID:122540	BioGRID:115691	UBAC1		BioGRID	PubMed	Affinity Capture-Western; Reconstituted Complex; Two-hybrid
BioGRID:122540	BioGRID:121626	ZNF471		BioGRID	PubMed	Two-hybrid
BioGRID:122540	BioGRID:126074	ZNF648		BioGRID	PubMed	Two-hybrid

Figura 4.20: Tabla de interacciones NCBI Gene.

Interaction

Subunit

Interacts (via CARD domain) with BCL10 (via CARD domain) (PubMed:**21302310**).
Forms a complex with MALT1 and BCL10; resulting in the formation of a CBM (CARD14-BLC10-MALT1) complex (PubMed:**27113748**, PubMed:**27071417**).
Interacts with TRAF2, TRAF3 and TRAF6 (PubMed:**21302310**). 3 Publications

Binary interactions

TYPE	ENTRY 1	ENTRY 2	NUMBER OF EXPERIMENTS	INTACT
-- Select --	-- Select --			
BINARY	Q9BXL6-2	PRPF31 Q8WWY3	4	EBI-12114736, EBI-1567797
BINARY	Q9BXL6-2	ZNF648 Q5T619	3	EBI-12114736, EBI-11985915

Protein-protein interaction databases

BioGRID | 122540 14 interactors
IntAct | Q9BXL6 7 interactors
STRING | 9606.ENSP00000498071

Miscellaneous

RNAct | Q9BXL6 Protein

Figura 4.21: Tabla de interacciones de UniProt.

Switch View: Interactors 14 | Interactions 22 | Network | PTM Sites 1

Showing 1 to 22 of 22 interactions

Filter Interactions... ADV

Interactor	Role	Organism	Experimental Evidence Code	Dataset	Throughput	HTP Score	Curated By	More
BCL10	BAIT	H. sapiens	Two-hybrid	Bertin J (2001)	Low	-	BioGRID	
BCL10	HIT	H. sapiens	Affinity Capture-Western	Bertin J (2001)	Low	-	BioGRID	-
BCL10	BAIT	H. sapiens	Reconstituted Complex	Bertin J (2001)	Low	-	BioGRID	-
BCL10	BAIT	H. sapiens	Affinity Capture-Western	Howes A (2016)	Low	-	BioGRID	-
HECW2	BAIT	H. sapiens	Affinity Capture-MS	Lu L (2013)	High	-	BioGRID	-
HULC	BAIT	H. sapiens	Affinity Capture-MS	Wang C (2020)	High	-	BioGRID	
MALT1	BAIT	H. sapiens	Affinity Capture-Western	Afonina IS (2016)	Low	-	BioGRID	-
MALT1	BAIT	H. sapiens	Affinity Capture-Western	Howes A (2016)	Low	-	BioGRID	-
MAP3K1	BAIT	H. sapiens	Biochemical Activity	Charlaftis N (2014)	High	-	BioGRID	
PCGF1	BAIT	H. sapiens	Affinity Capture-MS	Oliviero G (2015)	High	-	BioGRID	-
PLK1	BAIT	H. sapiens	Affinity Capture-MS	St-Denis N (2015)	High	-	BioGRID	-

Figura 4.22: Algunas interacciones proteína-proteína obtenidas en BioGRID

4.6.5. Ejercicio 6.5

De la base de datos Uniprot, se pueden obtener las ubicaciones sub-celulares de la proteína analizada. Las mismas se pueden observar en la figura 4.23. En resumen, se puede ver que CARD14 es una proteína citoplasmática.

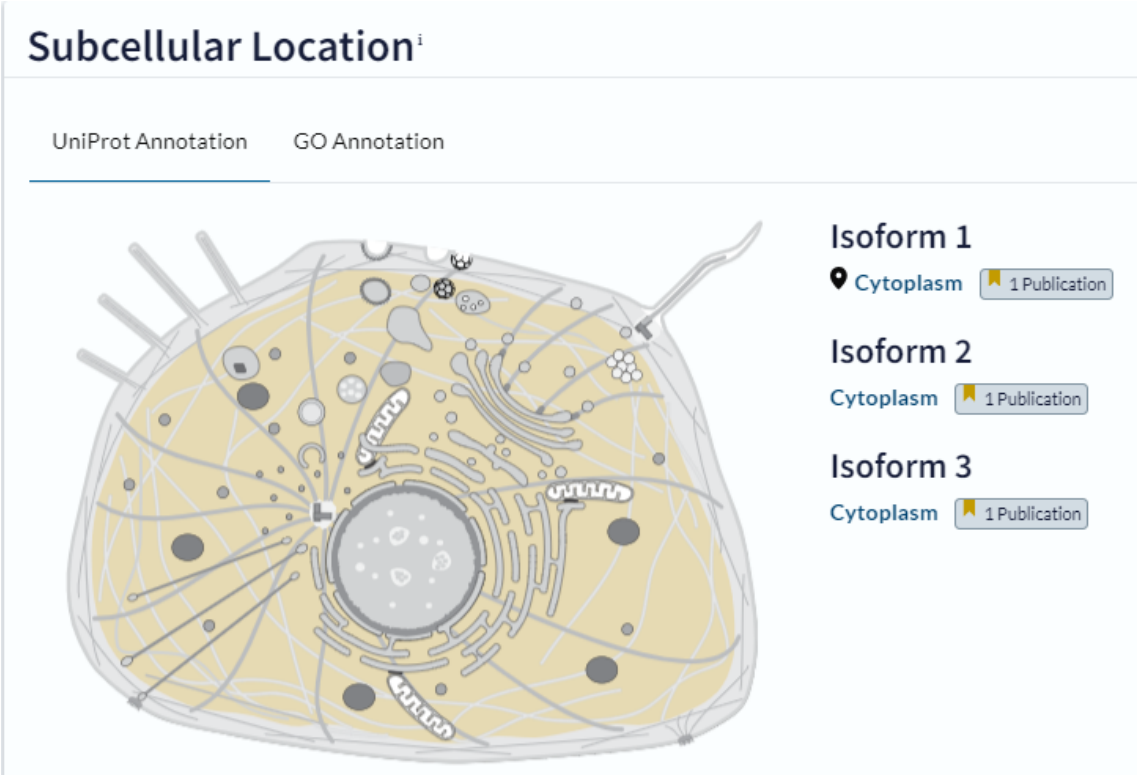

Figura 4.23: Ubicaciones sub-celulares de la proteína según Gene Ontology (UniProt).

Además, tambíen de UniProt, se pueden obtener las funciones moleculares y los procesos biológicos asociados a la proteína de interés. Esta información puede observarse en las Figuras 4.24 y 4.25, respectivamente.[16]

Molecular Function	CARD domain binding	Source:UniProtKB	1 Publication

Figura 4.24: Funciones moleculares encontradas en UniProt

Biological Process	activation of NF-kappaB-inducing kinase activity	Source:UniProtKB	1 Publication
Biological Process	apoptotic process	Source:UniProtKB-KW	
Biological Process	negative regulation of apoptotic process	Source:UniProtKB	1 Publication
Biological Process	positive regulation of NF-kappaB transcription factor activity	Source:UniProtKB	1 Publication
Biological Process	positive regulation of protein phosphorylation	Source:UniProtKB	1 Publication
Biological Process	tumor necrosis factor-mediated signaling pathway	Source:UniProtKB	1 Publication

Figura 4.25: Funciones biológicas encontradas en UniProt

Se pueden ver que tanto las funciones moleculares y biológicas encontradas en UNiProt coinciden con lo detallado anteriormente y con lo que se explicará a continuación.

4.6.6. Ejercicio 6.6

La principal vía metabólica en la que el gen CARD14 participa es la del NF-kappa B signaling pathway (NF-κB). Según lo investigado en KEGG, la vía NF-κB es un conjunto de factores de transcripción que funciona como dímeros y regulan genes involucrados en la inmunidad, reacción inflamatoria y supervivencia celular. El pathway general tiene 3 vías posibles: la canónica, la atípica ya la no-canónica.

En el caso de la canónica, es iunducida por el factor de necrosis tumoral alfa (TNF-alpha), la interleuquina 1 (IL-1) o subproductos bacteriales o virales. Se basa en la fosforilación de IkappaB-alfa mediada por IKK en Ser32 y 36, lo que lleva a su degradación, lo que permite que el dímero p50/p65 NF-kappa B entre en el núcleo y active la transcripción genética. La vía atípica es independiente de la IKK, sino que se basa en la fosforilación de IkappaB-alfa en Tyr42 o en residuos Ser en el dominio PEST de IkappaB-alfa. La vía no-canónica es desencadenada por miembros particulares de la superfamilia TNFR, como la linfotoxina beta (LT-beta) o BAFF. Implica la fosforilación de p100 mediada por NIK e IKK-alfa y su procesamiento a p52, lo que da como resultado la translocación nuclear de los heterodímeros p52/RelB. [17]

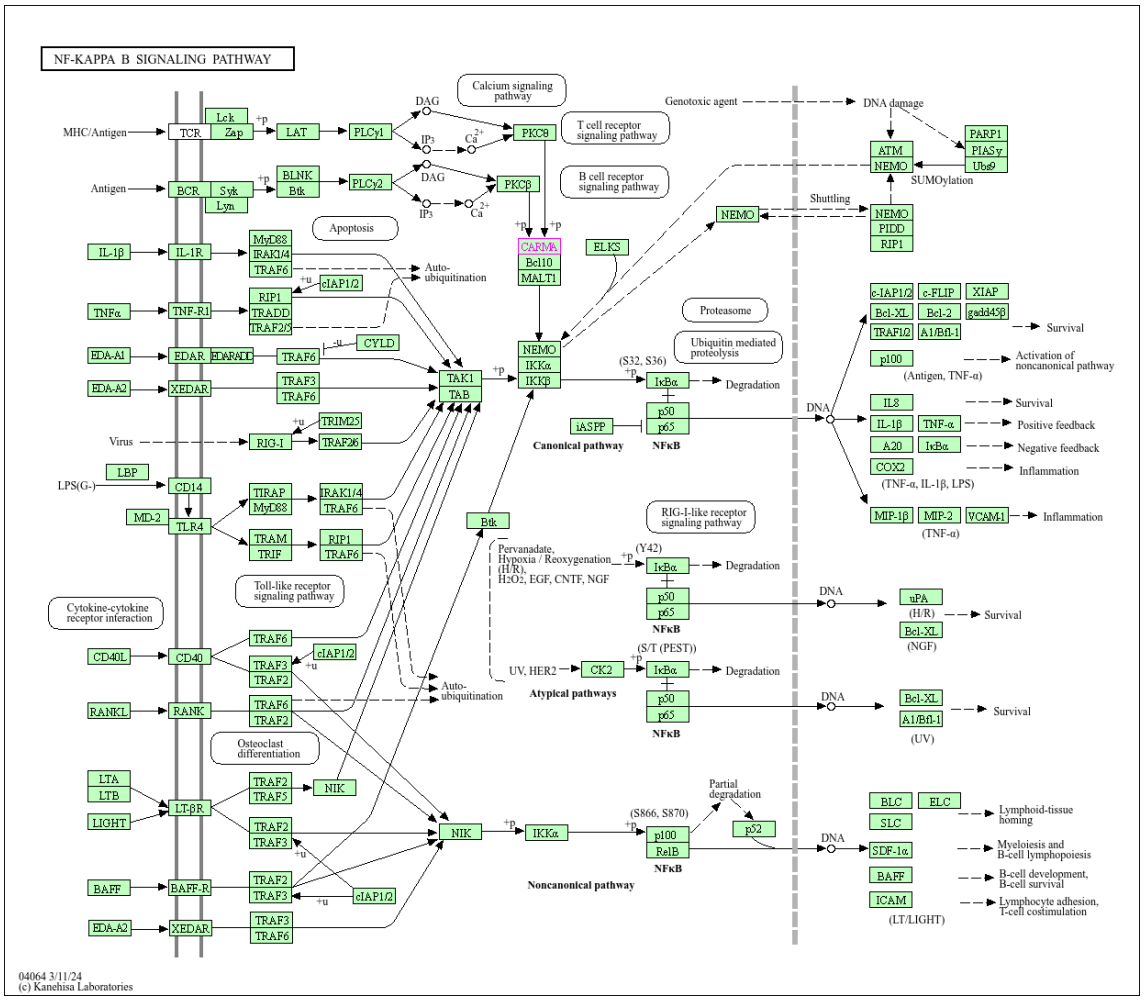

Figura 4.26: *Pathway* NF-κB

En el pathway de la Figura 4.26, se pueden ver los 3 tipos de pathways. La celda marcada en rosa es la que cotniene al gen CARD14 que tiene influencia en el pathway canónico. Esto es porque actúa como regulador específico de la función BCL10, la cual influye directamente en la IKK, que es uno de los activadores principales de la via canónica del pathway NF-κB, como se explicó anteriormente..

4.6.7. Ejercicio 6.7

En el caso del gen CARD14 de psoriasis, al buscar las variantes genéticas en dbSNP se eligió la variante **rs1046832**, que se encuentra en el cromosoma 17, posición 80209597[18]. Esta variante tiene la particularidad que intercambia las A por G. En la población la variante tiene una frecuencia oscilante entre 0,45 y 0,48. Observando valores más precisos de estudios globales, se ven las frecuencias 0,4507

(1000G), 0,460437 (TOPMED) y 0,473459 (GnomAD)

El grupo étnico más afectado parece ser el asiático, específicamente el japonés, con un promedio de 69 % de casos con la variante del gen en 2 estudios.

Referencias

[1] I. Michalek, B. Loring, and S. John, "A systematic review of worldwide epidemiology of psoriasis," *Journal of the European Academy of Dermatology and Venereology*, vol. 31, no. 2, pp. 205–212, 2017.

[2] F. Capon, M. Munro, R. Trembath, and J. Barker, "Searching for the Major Histocompatibility Complex Psoriasis Susceptibility Gene," *The journal of investigative dermatology/Journal of investigative dermatology*, vol. 118, pp. 745–751, 5 2002.

[3] F. Enlund, L. Samuelsson, C. Enerbäck, A. Inerot, J. Wahlström, M. Yhr, Torinsson, T. Martinsson, and G. Swanbeck, "Analysis of Three Suggested Psoriasis Susceptibility Loci in a Large Swedish Set of Families: Confirmation of Linkage to Chromosome 6p (HLA Region), and to 17q, but not to 4q," *Human heredity*, vol. 49, pp. 2–8, 1 1999.

[4] L. Samuelsson, F. Enlund, Torinsson, M. Yhr, A. Inerot, C. Enerbäck, J. Wahlström, G. Swanbeck, and T. Martinsson, "A genome-wide search for genes predisposing to familial psoriasis by using a stratification approach," *Human genetics*, vol. 105, pp. 523–529, 11 1999.

[5] J. Romaní and A. Casulleras, "Psoriasis y pitiriasis versicolor: juntas pero no revueltas," *Actas dermo-sifiliográficas/Actas dermo-sifiliograficas*, vol. 110, pp. 317–318, 5 2019.

[6] IVAMI, "Pruebas genéticas: Pitiriasis rubra pilaris familiar (familial pityriasis rubra pilaris, gen i: Card14)," 2024. Accessed: 2024-06-28.

[7] NCBI, "Card14 caspase recruitment domain family member 14." = https://www.ncbi.nlm.nih.gov/gene/79092summary.

[8] MedlinePlus, "Card14 gene." https://medlineplus.gov/genetics/gene/card14/, 2024. Accessed: 2024-06-26.

[9] IBM, "Agile explorer." https://skills.yourlearning.ibm.com/.

[10] "Notion." https://www.notion.so/.

[11] N. C. for Biotechnology Information, "Homo sapiens card14." https://www.ncbi.nlm.nih.gov/nuccore/?term=Homo+sapiens+CARD14, 2024.

[12] T. B. Project, "Bio.seq.seq.translate - biopython 1.75 api documentation." https://biopython.org/docs/1.75/api/Bio.Seq.html#Bio.Seq.Seq.translate, 2019. Accessed: 2024-04-28.

[13] B. Contributors, "Bio.align.applications - biopython 1.75 api documentation." https://biopython.org/docs/1.75/api/Bio.Align.Applications.html, 2019. Accessed: 2024-06-28.

[14] UniProt, "Q9bsl1 · ubac1_human." https://www.uniprot.org/uniprotkb/Q9BSL1/entry.

[15] UniProt, "Q9udy8 · malt1_human." https://www.uniprot.org/uniprotkb/Q9UDY8/entry.

[16] UniProt, "Card14 functions." https://www.uniprot.org/uniprotkb/Q9BXL6/entry#function.

[17] KEGG Pathway Database, "Nf-kappa b signaling pathway." https://www.genome.jp/entry/map04064, 2024. Accessed: 2024-06-26.

[18] N. dbSNP, "Reference snp (rs) report, rs1046832." https://www.ncbi.nlm.nih.gov/snp/rs1046832, Sep 2022.