ESCAPE Y ANALISIS DE LAS SECUENCIAS DE RETROTRANSPOSONES LTR EN EL GÉNERO *SOLANUM*.

Trabajo Final de Grado
del alumno

Agustin Caiciia Massello

Este trabajo ha sido presentado como requisito
para la obtención del título de

Licenciada en Genética

Carrera

Licenciatura en Genética

Escuela de Ciencias Agrarias, Naturales y Ambientales
Universidad Nacional del Noroeste de la Provincia de Buenos Aires.

Pergamino, 20 de Noviembre de 2020

ESCAPE Y ANALISIS DE LAS SECUENCIAS DE RETROTRANSPOSONES LTR EN EL GÉNERO *SOLANUM*.

Trabajo Final de Grado

del alumno

Agustin Caiciia Massello

Aprobada por el Tribunal Evaluador

Susana M. Pistorale	Gabriela P. Fernández	Rolando Rivera
Evaluador/a	**Evaluador/a**	**Evaluador/a**

Andres E. Lavore	Diego H. Sanchez
Co-Director/a	**Director/a**

Escuela de Ciencias Agrarias, Naturales y Ambientales,
Universidad Nacional del Noroeste de la Provincia de Buenos Aires

Pergamino, 20 de Noviembre de 2020

INDICE

INTRODUCCIÓN

(a) Transposones y evolución

Los transposones o elementos transponibles (en inglés, *transposable elements* -ETs-) son secuencias de ADN que se pueden movilizar dentro de los genomas de manera auto-suficiente, proceso que se denomina transposición (Grandbastien, 2015). Estos elementos evolucionaron como parásitos intra-genómicos 'egoístas', y pueden causar perjuicios a la estabilidad del genoma hospedador en virtud de su naturaleza mutagénica insercional. Clásicamente, en eucariotas, se diferencian dos tipos de ETs dependiendo de su mecanismo de movilización: de clase I (retrotransposones) que se movilizan replicativamente por "copiado y pegado", donde el elemento genera intermediarios transcripcionales (esto es, moléculas de ARN) que luego originan ADN a través de transcripción-reversa, finalmente insertándose en otra coordenada genómica (Ma et al., 2004); y de clase II (ADN-ETs) por "cortado y pegado", donde la secuencia del elemento se escinde de una coordenada genómica para re-insertarse en otra (Pierce, 2006).

Los ETs presentan secuencias funcionales, que no solo contienen áreas regulatorias y marcan los límites de los elementos, sino que también codifican enzimas necesarias para completar el ciclo de vida transposicional permitiendo que éstos se copien y se vuelvan a insertar en el genoma hospedador. La evolución de ETs sigue un modelo denominado de 'ráfaga y decaimiento', lo que significa que una familia de ETs puede multiplicarse rápidamente en un genoma, seguido de períodos con replicación relativamente baja (Lodish et al., 2005). Luego de una ráfaga, la mayoría de las copias de un ETs acumulan errores de manera independiente por deriva genética. Este decaimiento es debido al efecto acumulativo de mutaciones puntuales, inserciones anidadas, deleciones, e inserciones y deleciones cortas (*indels*). Como resultado, los ETs se vuelven cada vez más fragmentados y mutados con el tiempo, hasta que finalmente forman parte de las secuencias conocidas como *'junk'* -que son simplemente restos de secuencias de ETs que quedan en el genoma sin capacidad de transponerse-. Entre dos especies de hospedadores que divergieron de un antecesor común, las diferencias en las secuencias de ETs ocurren de manera independiente, acumulándose en función del tiempo. A partir de esto se pueden reconocer cuales fueron los polimorfismos acumulados entre los ETs de las dos especies de hospedadores, y datar cuándo sucedió la divergencia de sus secuencias.

Con esta estrategia se puede inferir la “edad” de ETs particulares; esto es, estimar el tiempo transcurrido desde la inserción transposicional (Maumus & Quesneville, 2016). Además de otros mecanismos, como la duplicación del genoma completo por poliploidía y grandes duplicaciones segmentarias, los ETs son los principales determinantes del tamaño del genoma de las plantas, que es muy variable.

(b) Silenciamiento de ETs

En virtud de la naturaleza mutagénica de los ETs en los genomas hospedadores, han evolucionado mecanismos de supresión. El silenciamiento de las secuencias de ETs se puede dividir en dos grandes categorías:

(i) El silenciamiento génico transcripcional (del inglés *transcriptional-gene-silencing* -TGS-); donde el silenciamiento epigenético de ETs se lleva a cabo principalmente a través de un proceso conocido como la metilación del ADN dirigida por ARN (del inglés *RNA-dependent-DNA-methylation* -RdDM-) (Fultz et al., 2015). La vía de RdDM utiliza pequeños fragmentos de ARNs de interferencia (siARNs) de 24 nucleótidos (nt) para guiar modificaciones epigenéticas secuencia-especificas, que eventualmente conducen al silenciamiento transcripcional de ETs. Estos siARNs son producidos a partir de transcriptos no codificantes por la ARN polimerasa Pol-IV. En plantas, la biogénesis de siARNs de 24 nt se da por actividad ‘*dicer’* de la enzima *Dicer-like* en su homologo 3 DCL3, y luego se cargan en proteínas *Argonaute* AGO4 y AGO6. Para dirigir la maquinaria de silenciamiento hacia áreas con ET, los siARNs de 24 nt se asocian con las proteínas Argonauta y luego forman un complejo con la ARN polimerasa Pol-V, que eventualmente media la metilación de citosinas y se marcan histonas represivas (Bousios & Gaut, 2016).

(ii) El silenciamiento génico post-transcripcional (del inglés *post-transcriptional-gene-silencing* -PTGS-); cuyo mecanismo actúa una vez realizada la copia del ARN mensajero a través de secuencias de miARN (*micro-ARN*) y siARN los cuales reconocen a los transcriptos de ETs y conducen a su degradación mediada por proteínas ARNasa (Cerutti & Casas-Mollano, 2006). Este segundo grupo de formas de silenciamiento ocurre a través de fragmentos de ARN de aproximadamente 21nt. Su biosíntesis es mediada por factores clave como la polimerasa de ARN RDR6, que convierte el mARN monocatenario (ssRNA, del inglés *single strand* RNA) en ARN de doble hebra (dsARN); y luego proteinas ‘*dicer*’ como DCL2 y DCL4, que escinden el dsARN en 21 - 22nt siARNs. Estos últimos son finalmente cargados en la proteína

Argonauta AGO1 para guiar la degradación de los mensajeros por complementariedad de secuencia. Esta regulación post-transcripcional puede no solo degradar un gran número de mARN pertenecientes a ETs, sino también podría desencadenar la 'iniciación' del TGS, y, por lo tanto, proporcionar un punto de entrada en RdDM (Bousios & Gaut, 2016).

(c) Retrotransposones LTR: características y ciclo de vida

Los retrotransposones LTR (repetición terminal larga, del inglés *long-terminal-repeat*) incluidos en la clase I tienen similitudes estructurales y funcionales con los retrovirus y son los ETs predominantes en las plantas (Grandbastien, 2015). Se replican a través de un intermediario de ARN que luego origina ADN hijo bicatenario por actividad transcriptasa reversa (Wicker et al., 2007). Las secuencias de los LTRs contienen *cis-elements* regulatorios que típicamente flanquean una zona codificante, con al menos dos marcos abiertos de lectura (del inglés *open-reading-frames* -ORFs-) canónicos. Los ORFs codifican para enzimas necesarias en el ciclo de vida de estos ETs: una proteína estructural llamada *structural RNA-binding protein* GAG y una poliproteina POL (Grandbastien, 2015). Estas proteínas incluyen la proteína de unión a ARN estructural de GAG, y dentro de POL la proteasa requerida para escindir unidades de proteína funcionales de la poliproteína GAG-POL, la transcriptasa reversa con actividad ARNasa H asociada (RT / RH) que asegura la síntesis de la copia de ADN hija secundaria del intermediario de ARN, y finalmente la integrasa responsable de insertar en el genoma la copia hija de ADN bicatenario (Devos et al., 2002) (Figura 1). Otras características típicas incluyen secuencias cortas de PBS (del inglés *priming-binding-site*, sitio de unión de cebador) y PPT (del inglés *poly-purine-track*, tracto de polipurina), que están implicadas en la síntesis de la cadena de ADN hija durante la transcripción reversa (Ma et al., 2004). Al igual que la mayoría de los ETs, las inserciones de retrotransposones LTR están limitadas por repeticiones directas cortas (del inglés *tándem-site-duplications* –TDS-) generadas tras la inserción en el sitio anfitrión. Los retrotransposones LTR están enriquecidos en regiones de heterocromatina pericentroméricas de los genomas huéspedes, siendo los principales componentes de ADN de las plantas con flores. Por ejemplo, aproximadamente el 25% del genoma de *Oryza sativa* (arroz), el 42% del genoma de *Glycine max* (soja) y el 75% del genoma de *Zea mays* (maíz) se componen de retrotransposones LTR (Ou & Jiang, 2018).

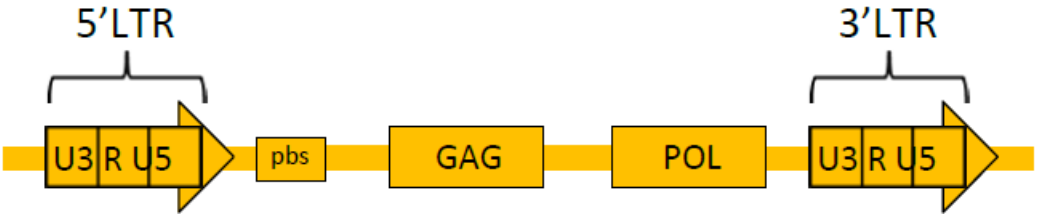


Figura 1: Estructura general de retrotransposososones LTR

Muestra las regiones que codifican proteínas estructurales y enzimáticas, y la disposición general de los LTR flanqueantes (Drost & Sanchez, 2019).

Es importante destacar que la similitud de secuencias entre LTRs son útiles para tipificar la edad de la inserción de un retrotransposon LTR. Esto es así porque en el instante de inserción, ambos LTRs son idénticos -en virtud de los mecanismos de amplificación y movilización descriptos anteriormente-. Sin embargo, en función del tiempo evolutivo de permanencia en el genoma, los LTRs de un mismo elemento van independientemente acumulando mutaciones por deriva genetica. Por ende, cuanto menor es la diferencia de secuencia entre LTRs, el retrotransposon es más reciente o moderno; esto es, más actual su transposición (Drost & Sanchez, 2019).

(d) El género *Solanum* como hospedador de ETs

Las solanáceas son una familia de angiospermas particularmente interesante, no sólo porque incluyen muchas especies de cultivos importantes (por ejemplo, papa - *Solanum tuberosum*, tomate - *S. lycopersicum*, berenjena - *S. melongena*, pimientos dulces y picantes - *Capsicum spp*., Tabaco -*Nicotiana spp*) y ornamentales (por ejemplo, *Petunia spp.*), sino también varios taxones utilizados como sistemas de modelos biológicos (por ejemplo, *Nicotiana spp*., *Solanum spp., Petunia spp., Datura spp*.). Un marco taxonómico y filogenético ahora está disponible para la familia en la web (http://www.solanaceaesource.org). Para *Solanum* en sí, la filogenia más reciente incluyó solo 102 (7,7%) del total de casi 1.325 especies en el género. Debido a que se han publicado varios estudios filogenéticos en varios niveles taxonómicos

en *Solanaceae*, ahora hay una gran cantidad de nuevos datos de secuencia disponibles para un análisis más amplio a nivel familiar (Ou & Jiang, 2018).

La completa re-secuenciación de especies del género *Solanun* (Consortium, 2012; Bolger et al., 2014; *Tomato Genome Sequencing* et al., 2014) demostró un extensivo contenido de elementos repetitivos en estos genomas, lo que facilita estudios de genómica comparativa entre familias de ETs. En el genoma de tomate, los retrotransposones LTR recientes y los elementos relativamente antiguos presentan diferentes patrones de distribución, y las tasas de recombinación genética (GR) y los tiempos de inserción de los retrotransposones LTR están negativamente relacionados entre sí (Xu & Du, 2014). Estos datos también muestran que la variación estructural de los retrotransposones LTR varía con diferentes regiones genómicas, tales como regiones génicas, regiones intergénicas y regiones asociadas con genes; lo que indica la importancia de GR y la selección en la remodelación de la secuencia del genoma del tomate. Justamente, casos paradigmáticos en tomate, han demostrado la capacidad de ciertos retrotransposones LTR de re-organizar las funciones génicas, cambiando el fenotipo expresado por el hospedador (Xu & Du, 2014). Por ejemplo, el *locus* SUN de tomate involucra varios genes algunos implicados en la forma del fruto, los cuales han sido duplicados, re-organizados, re-alocados y/o de-regulados por miembros de retrotransposones LTR de la familia Rider (Xiao et al., 2008; Jiang et al., 2009).

En este trabajo final de grado se explora por genómica comparativa las familias de retrotransposones LTR presentes en *Solanum pennellii* y *Solanum lycopersicum*. La familia *Solanum* presenta buenos modelos vegetales para revelar fenómenos de evolución de ETs, gracias a que se dispone de genomas secuenciados de buena calidad en varias de sus especies, las cuales poseen grandes genomas con muchos retrotransposones LTR potencialmente activos. A partir del estudio de secuencias divergentes, se realizan inferencias de cómo se diversificaron y evolucionaron los mecanismos de escape al silenciamiento en familias de ETs durante los procesos de especiación en genotipos filogenéticamente muy cercanos.

HIPÓTESIS

Los mecanismos de escape al silenciamiento en retrotransposones LTR evolucionan rápidamente durante los procesos relacionados a la especiación en genomas hospedantes. Los cambios en las secuencias de ETs pueden ser reconocidos a través de genómica comparativa al explorar hospedantes filogenéticamente cercanos.

OBJETIVOS

(a) Objetivo general

Usar herramientas bioinformáticas para entender cómo evolucionan las secuencias responsables de la activación transcripcional en retrotransposones LTR del género *Solanum*, que revelarían mecanismos de escape a la maquinaria de silenciamiento.

(b) Objetivos específicos

- Adquirir conocimientos y entrenamiento en el análisis informático de genómica funcional: aprendizaje de herramientas en *CLI* Linux aplicadas al análisis de secuencias y *next-generation-sequencing*.
- Anotar *de-novo* retrotransposones LTR en especies re-secuenciadas del género *Solanum*, utilizando algoritmos de última generación y validación con *pipelines* independientes.
- Evaluar la divergencia familiar de retrotransposones LTR entre especies del género *Solanum*, utilizando herramientas de *clustering* y alineamiento.
- Estimar la expresión de retrotransposones LTR a través del mapeo y conteo de experimentos de ARN-Seq en especies del género *Solanum*.
- Modelar la micro-evolución de las secuencias responsables del control transcripcional en retrotransposones LTR por genómica comparativa, infiriendo mecanismos evolutivos de la activación transposónica tejido-específica.

MATERIALES Y MÉTODOS

(a) Obtención de secuencias candidatas de transposones

Los datos iniciales resultaron de la anotación de retrotransposones LTR, obtenidos a través del análisis con LTR Harvest (Ellinghaus et al., 2008). Los parámetros del *software* utilizados se presentan en el Anexo I.

Los genomas utilizados para extraer los retrotransposones LTR fueron los de las especies *Solanum lycopersicum* (tomate doméstico relativamente nuevo en términos genéticos, usándose la versión 3.0 del *assembly* disponible en www.solgenomics.com), y *Solanum pennellii* (especie salvaje emparentada a tomate, usándose la versión 2.0 del *assembly* disponible en www.solgenomics.com). El empleo de especies relacionadas permitirá inferir si los ET estuvieron activos a partir del momento de especiación, analizando la filogenia de los ET de ambas especies. La salida de esta herramienta es en formato tabulas donde se pueden ver los resultados de la anotación, que puede ser extraído en un archivo de texto para poder manipularlos. Los resultados se convirtieron a forma GFF3, que se usó como archivo base para trabajar. Posteriormente se convirtió en un formato BED, el cual no sólo es un formato más simple y fácil de manipular sino que permite la rápida obtención de las secuencias FASTA usando la herramienta *Bedtools* (Quinlan & Hall, 2010) *y se* contabilizaron los ETs y *clusters*.

(b) Agrupamiento de elementos similares

De la secuencia de los ETs extraídos, se separan dos elementos genéticos esenciales para nuestro análisis: los 5LTR (secuencia al inicio de los retrotransposones LTR en el extremo 5') y las secuencias INNER (secuencia entre los LTRs). Una vez corroborado que los archivos eran correctos, es decir, que los archivos no estaban corruptos ni presentaban fallos, se generó un archivo general donde se unieron los archivos de las dos especies, manteniendo la separación en archivos diferentes las secuencias INNER, 5LTR y las secuencias completas (el ET completo). Se comprobó que estuviesen presentes todos los elementos para el *input* de secuencias antes de aplicar el *clustering* mediante *Vsearch* (Rognes et al., 2016).

Antes de realizar el *clustering,* se randomizó las secuencias para no generar ningún sesgo y para que el resultado de los *clusters* sea más consistente. Una vez

terminado el proceso se tomó el *output* y se realizó el *clustering* por separado de las secuencias 5LTR e INNER con las siguientes especificaciones:

El *output* del *Vsearch* es un archivo ".txt" con una configuración particular; se convirtió a ".gff3" mediante un *script* de *Python* siendo éste una mejor forma de trabajar las secuencias resultantes de los *clusters* ya agrupados. Los archivos ".gff3" se fueron creando de un archivo por *cluster* de ETs para una separación y descarte más fácil.

En simultáneo, de los *output* del total de los posibles ETs extraídos por LTR Harvest, se extrajeron las secuencias de ADN que los componen mediante la herramienta *Bedtools* (Quinlan & Hall, 2010). Para esto se tomaron como entrada los archivos BED de los ETs y los genomas generados en fasta de ambas especies, generando fastas con todos los ETs secuenciados.

Teniendo ambos archivos, se pudo obtener los *clusters* separados y las secuencias de éstos en un archivo FASTA. Para lograrlo, se programó en *Python*, y a partir del *tag_id* se extrajo el segmento secuenciado que correspondía al transposón.

Se contabilizó el número de ETs obtenidos hasta ese momento, tomándolo como un punto de control, con la finalidad de asegurarse no perder datos importantes. Se descartaron los *clusters* conteniendo menos de 4 ETs mediante un *script* de *Python*, ya que estimamos no ofrecerían datos suficientes para establecer patrones de secuencia fiables por genómica comparativa. Los *clusters* con más de 4 ETs se separaron en carpetas distintas, quedando pocos *clusters* para analizar; dando un número más fácil de manejar.

(c) Filogenia

Una vez que se obtuvieron y definieron los *clusters* a utilizar, se realizaron los alineamientos a partir de los cuales fueron generados los árboles filogenéticos. De esta manera, se identificaron aquellos ETs que probablemente estén más emparentados. Se utilizó *MUSCLE* (Edgar, 2004) para hacer los alineamientos, con los parámetros que vienen por defecto.

Pero como son varios *clusters* en archivos separados, se utilizó un *script* de *Python* para separar las secuencias por *clusters,* alinearlos con *MUSCLE* (Edgar, 2004), y finalmente armar otra carpeta con los *cluster* elegidos y alineados.

En el armado de los árboles se usó el formato *phylip*, que es un tipo de archivo de descripción del árbol filogenético, utilizado por el *Crustal W* y posee un formato que casi cualquier *sofware* lo puede leer. Para diseñar los árboles se utilizó el comando "-*tree2*" que especifica el modo de interacción entre elementos. Las especificaciones del comando son: "*Matriz de distancia por identidades en pares desde la alineación múltiple actual, alineación progresiva según el nuevo árbol, repita hasta la convergencia o el número máximo especificado de veces*".

Por último, se especificó la metodología de reconstrucción filogenética para la obtención del árbol, siendo ésta *neighborjoining* (vecino más cercano), donde la genealogía se detecta progresivamente según la menor distancia genética; agregando ramas según sean más cercanos (mayor consenso en sus secuencias) o lejanos.

Con los árboles resueltos, se necesitó otro *software* para visualizarlo gráficamente, ya que el formato de salida solo es descriptivo y no sirve para el análisis gráfico e intuitivo. Se recurrió al software MEGA (Kumar et al., 2018) para lograr visualizar y analizar las relaciones filogenéticas, especificándole en el test de filogenia en 1000 *bootstraps*.

Los árboles debieron ser armados dos veces, la primera para capturar cuales de los ET se encontraban en sentido reverso-complementario. Para tabular todos los ETs en el mismo sentido, primero se obtuvieron los árboles en formato bimodal y se confeccionó un archivo de Excel donde se especificó la dirección de transcripción inferida de cada ETs en aquellos *clusters* de interés. Usando un comando de *bio-python*, y como guía esta base de datos en Excel, se invirtieron las secuencias que se encontraban en reverso-complementario. Ya con las secuencias en el mismo sentido, se volvió a realizar el armado de los árboles (con las mismas especificaciones desde el alineamiento hasta la visualización de los mismos), obteniendo los árboles correctos. Una vez obtenidos los árboles de las secuencias INNER, se realizó el segundo armado del protocolo para las secuencia de los 5LTR, desde la obtención de *clusters* hasta los árboles.

(d) Transposones *de novo*

Una vez corregida la orientación de los ETs se pasó a analizar cuáles de éstos estaban realmente escapando y cuáles fueron falsos positivos. Para esto, inicialmente se revisó la similitud de los extremos LTR obtenida del LTR Harvest (Ellinghaus et al.,

2008), reconociendo los transposones más modernos De los árboles con todos los ETs en la misma dirección se comprobó la familias usando BLAST de proteínas de la secuencia INNER. Finalmente, también se comparó la anotación de los genes y los *reads* de secuenciación desde el *software* Seqmonk, Los árboles se utilizaron para ver a las familias de ETs salido de una ráfaga de transcripción y con una similitud LTR alta, que dio indicios de probable movilización recientemente. Lo siguiente fue observar si la transcripción del transposón está dada por un gen cercano y/o si éste está influenciado por algún gen; en otras palabras, si un ETs está realmente escapando del silenciamiento o no, y con las búsquedas de proteínas en el BLAST se verificó que realmente es un transposón al encontrarse una alta similitud con secuencias proteicas que caracterizan los ETs.

Para la confirmación final de que los *clusters* y ETs realmente escapan del silenciamiento génico, se realizó un conteo de los *reads* de la secuenciación. Éstos fueron estimados mediante Salmon (Patro et al., 2017), cuya salida es en transcriptos por millón (TPM); como su nombre lo indica, es la proporción de transcriptos del fragmento escalado a un millón de mensajeros. Los ETs que superan 1 TPM en cualquier tejido se aceptaron como transcripcionalmente activos, aunque la importancia evolutiva sería mayor en el caso de aquellos con actividad en tejido meristemático. Esto es así, ya que en él se acarrea la línea germinal, y por ende la activación de ETs en este tejido puede hacerlos heredables, siendo potencialmente detectados en las generaciones siguientes. Una vez obtenida una lista de ETs aceptados como activos, y eligiendo aquellos más informativos, se realizó un análisis de los promotores presentes en los 5LTR para saber si éstos dan un indicio de porqué escapan.

Una vez estipulado el ET que escapa, se procedió a identificarlos. Para lograr ello, primero se realizó un BLAST de las secuencias en Gypsy Database (Llorens et al., 2010) para identificar el grupo mayor al cual pertenecen (Llorens et al., 2010) y a la familiar de pertenencia, se realizó el BLAST de las secuencias originales. Inicialmente se investigaron como base comparativa -*query*- familias ya descriptas de *S. lycopersicum* como MESSI y Rider a modo de control positivo. Para hacer la comparación anterior se utilizó Vsearch (Rognes et al., 2016) con la función de *usearch*.

(e) Análisis de *cis-elements*

Se ha demostrado en trabajos anteriores que los *cis-elements* ubicados en secuencias LTR tienen una fuerte influencia en la activación de los retrotransposones LTR (Galindo-González et al., 2017). En base a esto, se realizó un análisis comparativo de *cis-elements* entre elementos de distintos *clusters* de interés, usando como control las secuencias conocidas de Rider cuyos *cis-elements* ya han sido caracterizados (Benoit et al., 2019). En primer lugar, se realizó un búsqueda de similitud mediante el software de SIGNAL SCAN en la plataforma PLACE (Higo et al., 1999) para encontrar todos los *cis-elements* presentes en los ET de cada *cluster*, prestando especial atención que en el *cluster* de Rider estén presentes aquellos típicos ya descriptos. Una vez encontrados estos *cis-elements*, se realizó un análisis comparativo para evaluar cuales son comunes según familia en *clusters* informativos sobre escape al silenciamiento; esto es, aquellos *clusters* que presentan un número significativo de ETs con esta cualidad. Se analizó la proporción de aparición de estos *cis-elements* en cada *cluster,* y el promedio de aparición por ETs individual.

Adicionalmente se analizaron de manera similar los *cis*-elemento de dos *clusters* conteniendo la familia MESSI (y una MESSI-like), buscando en especial *cis-elements* relacionados con auxinas y expresión en meristemas, ya que este tipo de ETs suele escapar del silenciamiento en dicho tejido (Sanchez et al., 2019). De los que estaban presentes en alguno de los dos *clusters* relacionados a MESSI, se revisó que porcentaje se encontraba en el resto de los grupos relacionados a la superfamilia *gypsy.*

RESULTADOS

De los genomas de las especies emparentadas de tomate *S. lycopersicum* y *S. pennellii* (familia de *Solanaceas*), se recuperaron los LTR retrotransposones modernos utilizando el software LTR Harvest (Ellinghaus et al., 2008):

- retrotransposones LTR en *S. pennellii*: 2511
- retrotransposones LTR en *S. lycopersicum*: 4462
- retrotransposones LTR totales: 6973

Dado el tiempo de especiación entre estos dos genomas (~2-3 millones de años según Sarkinen et al. 2013), retrotransposones con una similitud entre 5’ y 3’ LTR >97% (aproximadamente 1.5 millones de años desde la inserción) demostrarían

familias de elementos transposicionalmente activos *a posteriori* del evento de especiación.

En base a esto obtuvimos los retrotransposones LTR modernos que se encontraban en ambas especies, un total de 6.973. Desde este punto generamos 3 grupos que analizamos en paralelo: la sección INNER (entre-LTRs), 5LTR (LTR izquierdo) y ETs (elementos completos). Estos ETs fueron agrupados según su homología de secuencia de ADN en la sección INNER, para reconocer relaciones familiares, además de potencialmente descartar falsos positivos. El resultado fue en total 4444 ETs completos (esto es, sin bases ambiguas 'N' que provienen de la secuenciación incompleta del genoma de referencia actual), repartidos en 1677 *clusters* que representarían familias/subfamilias de retrotransposones LTR.

- Número de *clusters*: 1677
- Número total de ETs completos: 4444 (comparación con LTR Harvest)
- Número de ETs incompletos: 2529 (comparación con LTR Harvest)

De estos 1677 *clusters*, muchos estaban conformados por muy pocos ETs o elementos únicos que no permiten llevar a cabo genómica comparativa, por lo que eliminamos aquellos con menos de 4 elementos, quedando un total de 120 *clusters*. Por la misma razón, también eliminamos los *clusters* que tenían ETs de una única especie hospedante. El resultado final de esta serie de filtros fue de 31 *clusters* potencialmente informativos.

Con el *software* Salmon (Patro et al., 2017), aquellos ETs transcripcionalmente activos se cuantificaron a través de los *reads* de transcriptomas. Usando transcriptomas de literatura obtenidos en distintos tejidos (hojas, flores y meristemas) (Sanchez et al., 2019), los ETs que sobrepasaban a 1 TPM en cualquier tejido se lo categorizaba como ET de escape. A continuación se muestra una tabla ya filtrada con los transposones que se consideraron activos (Tabla 1). Luego se realizaron árboles filogenéticos de aquellos *clusters* con elementos activos para ver cómo se relacionaban con los otros elementos de la familia; sin embargo, estos no siguieron ningún patrón de distribución que se perciba a simple vista.

Tabla 1: ETs que escapan al silenciamiento.

Length	Leaves(TPM)	Leaves(TPM)	Flowers(TPM)	Flowers(TPM)	Meristems(TPM)	Meristems(TPM)	cluster	clus_size	ID
4888	1.871705	2.467256	0	0	5.061632	7.803683	5	110	Sly_LTR_02_149
4658	2.893284	1.364076	0.436297	1.204271	2.955669	3.596538	5	110	Sly_LTR_02_154
4773	0.000001	1.345162	0.481449	0.000002	2.286126	6.248016	5	110	Sly_LTR_06_102
4514	0.499508	0.224148	0.389461	0.651122	1.076118	1.310212	5	110	Sly_LTR_07_207
4874	2.119444	0.877268	0.491107	0.299551	1.653078	3.300918	5	110	Sly_LTR_10_35
4603	0.63626	0.801712	0	0	3.596959	3.626412	5	110	Spenn_LTR_01_488
4870	0.926539	0.697584	0.691829	0.58349	1.926674	1.046758	5	110	Spenn_LTR_04_36
4596	0.162952	0.324207	0.069256	0.264653	1.826307	2.506648	5	110	Spenn_LTR_07_20
4540	0.393656	0.129087	0.439204	0.049741	1.712007	1.859422	5	110	Spenn_LTR_08_231
4870	1.508102	0.948795	0.116468	0.358025	1.668544	1.766326	5	110	Spenn_LTR_09_300
9392	2.582601	3.506601	1.333116	0.882639	2.743153	2.943331	20	47	Sly_LTR_03_151
9603	1.774695	1.495139	0.715744	0.772154	6.233353	7.661128	20	47	Sly_LTR_08_166
9488	2.195985	0.991861	0.609165	0.65679	5.099883	4.836767	20	47	Spenn_LTR_01_347
9669	6.301778	3.166622	0	1.980607	6.289664	4.807967	20	47	Spenn_LTR_01_408
9602	0.327136	0.50154	0.294762	0.274147	2.466313	2.08052	20	47	Spenn_LTR_04_183
9653	1.425637	1.47676	0	0.000331	2.513093	1.129646	20	47	Spenn_LTR_06_163
9654	7.152115	4.831472	0	0	5.295191	5.675978	20	47	Spenn_LTR_09_49
9662	1.433923	2.032335	0.198199	0.208967	4.254444	2.867162	20	47	Spenn_LTR_09_6
9548	0.87693	0.463473	0.129733	0.341222	1.468305	1.423458	20	47	Spenn_LTR_11_214
11627	3.444036	2.489333	0.528566	0.673076	4.451101	4.531587	53	24	Sly_LTR_02_167
12249	1.088206	0.810603	0	0.000006	3.717432	1.372184	53	24	Sly_LTR_02_177
11602	8.972283	6.837257	8.145274	6.592522	10.17166	9.078487	53	24	Sly_LTR_03_189
12170	2.5249	2.750905	1.486671	3.502601	7.30823	7.514999	53	24	Spenn_LTR_10_71
17296	5.382384	4.420619	4.046554	3.215354	10.39996	12.01633	152	9	Sly_LTR_01_155
20524	0.715125	0.736577	0.455741	0.47455	1.722309	1.921599	152	9	Sly_LTR_05_54
20553	5.903486	4.532895	6.815748	4.789926	11.30257	12.79294	152	9	Sly_LTR_05_91
19748	3.468738	3.131265	4.98364	3.779694	11.06872	10.93233	152	9	Sly_LTR_10_4
19071	1.064946	0.845254	0.808871	0.573925	1.101646	1.411304	152	9	Sly_LTR_10_235
20272	8.468002	7.397777	13.03654	12.57479	16.0332	17.80357	152	9	Sly_LTR_11_152
16869	6.239199	4.707264	7.765201	5.369651	11.83762	11.27073	152	9	Sly_LTR_12_143
21524	2.528696	1.477273	1.697155	1.049605	59.23977	61.66255	161	11	Sly_LTR_01_73
21091	0.544232	0.373406	0.604727	0.78811	9.631938	9.845933	161	11	Sly_LTR_01_91
24424	1.321068	0.477426	0.541492	0.418965	10.32353	9.722758	161	11	Sly_LTR_01_30
20021	1.66662	0.891278	1.787971	1.178435	29.84425	29.61447	161	11	Sly_LTR_01_315
21455	23.99659	13.19999	11.45217	13.06232	601.0744	528.7349	161	11	Sly_LTR_03_10
21089	2.246533	2.500967	1.345128	1.44553	185.5737	164.2741	161	11	Sly_LTR_03_187
20268	1.081579	0.448982	0.441939	0.541823	9.399033	10.98481	161	11	Sly_LTR_07_143
21867	5.54985	4.589808	21.95194	23.72674	64.47575	65.30775	161	11	Sly_LTR_08_214
20470	0.317077	0.398543	0.978794	0.472769	4.229559	4.769118	161	11	Sly_LTR_10_249
19559	0.334863	0.354175	2.118103	3.196563	18.68241	18.34587	161	11	Sly_LTR_10_262
19551	1.689144	1.351575	1.378333	1.816915	30.38167	27.77854	161	11	Sly_LTR_10_270
11150	0.650209	0.76195	0.468215	0.457758	2.691398	2.279313	216	9	Sly_LTR_01_71
11149	0.148595	0.19701	0.486861	0.311406	2.430611	2.334581	216	9	Sly_LTR_01_243
10622	1.411788	1.39121	1.900194	1.020338	4.491204	4.153799	216	9	Sly_LTR_02_98
10323	1.978971	1.983995	1.836885	1.681486	1.724095	2.121612	216	9	Sly_LTR_02_173
10414	3.340391	2.49897	1.896884	2.091823	6.29821	5.104746	216	9	Spenn_LTR_02_286
10407	0.746606	0.46404	0.764006	0.647737	1.676635	1.807313	216	9	Spenn_LTR_02_297
17546	0.982454	0.794714	2.450155	2.631785	15.68399	18.1922	220	6	Sly_LTR_01_74
20165	5.206687	3.554549	4.616221	2.994661	25.93953	29.19708	220	6	Sly_LTR_01_297
20387	1.877773	1.058162	5.774137	8.513593	19.29785	21.47365	220	6	Sly_LTR_01_338
20651	0.826467	0.495833	0.456406	0.278857	4.454278	5.320474	220	6	Sly_LTR_03_44
20606	0.347524	0.165682	1.196105	0.972481	5.034221	5.083153	220	6	Sly_LTR_10_257
21565	1.339305	0.885132	2.115389	1.685505	82.66637	77.44136	220	6	Spenn_LTR_06_90
10873	3.199284	2.156159	3.0219	3.120936	7.456155	8.178923	245	15	Sly_LTR_01_55
11299	2.387225	2.043696	2.666558	2.419802	5.863607	5.500194	245	15	Sly_LTR_12_100
10949	18.17103	22.56298	4.998728	6.74831	10.50335	8.23452	245	15	Spenn_LTR_01_465
10963	1.081139	1.100449	0	0.896266	1.536561	2.357614	245	15	Spenn_LTR_04_362
10953	4.721916	3.938049	1.922375	0.856463	8.024445	8.863458	245	15	Spenn_LTR_05_179
10979	1.5049	2.626191	0.565019	0.090101	3.540939	4.109856	245	15	Spenn_LTR_08_302
9928	28.2648	28.74968	3.9772	4.84088	7.561484	6.878655	245	15	Spenn_LTR_11_201
5208	6.434657	4.530856	2.26152	2.426072	3.708081	3.773547	279	9	Sly_LTR_04_164

Nota: Los TPM en Salmon muestran el nivel estimado de transcriptos que tiene cada uno de los ET dividido por tejido y en duplicado. El valor total de las columnas de TPM difiere porque en la tabla correspondiente no se muestran todos los datos, ya que para llevar a cabo esta estimación fue utilizada no solo los elementos transponibles sino también los genes. De esa manera seleccionamos a los que superan 1 TPM en meristema.

Esta estimación permite centrarnos en aquellas familias/sub-familias (*clusters*) con elementos que potencialmente pueden evadir el silenciamiento génico. El siguiente paso fue un BLAST de proteínas en la secuencia INNER, para confirmar proteínas típicas de los ETs como son GAG y/o POL, descartando cualquier secuencia que represente un falso positivo. Luego, con el *genome-browser* Seqmonk,

se revisaron los transcriptomas y la ubicación cromosómica en el genoma de las secuencias que escapaban, corroborando que éstos no sean -o no se estén expresados- por genes flanqueantes (Figura 2). Con este paso de corroboración, podemos estar seguros que los ETs reconocidos realmente se transcriben por sí mismos y no debido a que están cerca de un gen.

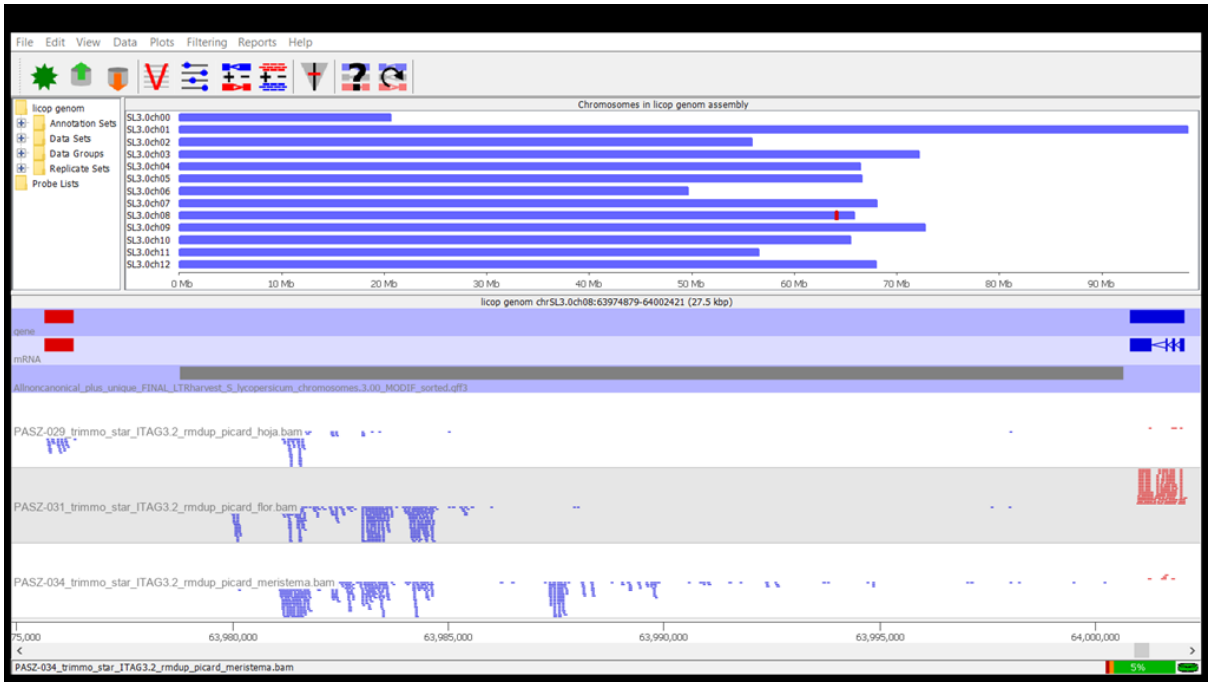
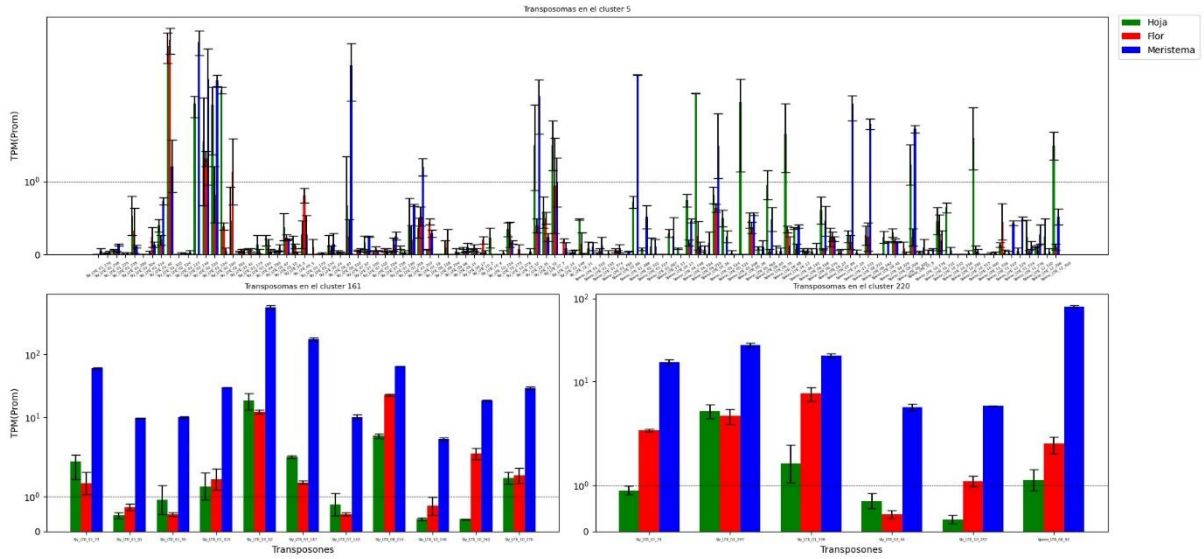


Figura 2: Extracto Seqmonk

En la figura se muestra un extracto del *genome-browser* Seqmonk, donde se visualiza un ET del *cluster* 161 (MESSI); se puede observar la acumulación de *reads* en el área de la anotación del transposón. En la imagen se ve la ubicación del TE en todo el genoma, y por debajo de esto se ve la anotación de los genes, RNA mensajero y el ET individual. Los *reads* son mostrando por debajo de las anotaciones por medio de 3 canales o gráficos donde se muestran los *reads* encontrados en 3 tejidos de interés: hoja, flor y meristema en orden descendiente.

En *S. lycopersicum*, Rider y MESSI son dos familias de ETs que ya se sabe poseen elementos modernos, y que al mismo tiempo pueden escapar al silenciamiento génico (Cheng et al., 2009; Sanchez et al., 2019). A modo de validación del análisis anterior, buscamos con BLAST aquellos *clusters* donde estas dos familias estuviesen representadas. Efectivamente, ambos fueron encontrados entre los *clusters* que seleccionamos como informativos, siendo el *cluster* 5 para Rider y los *clusters* 161 y 220 para MESSI (Figura 3a).

(a)



(b)

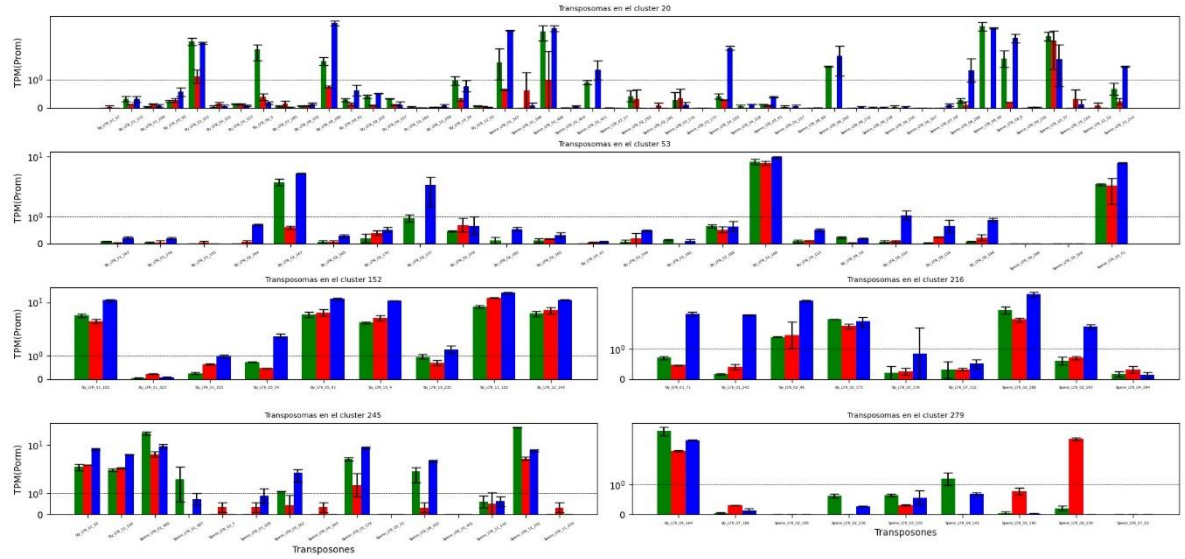


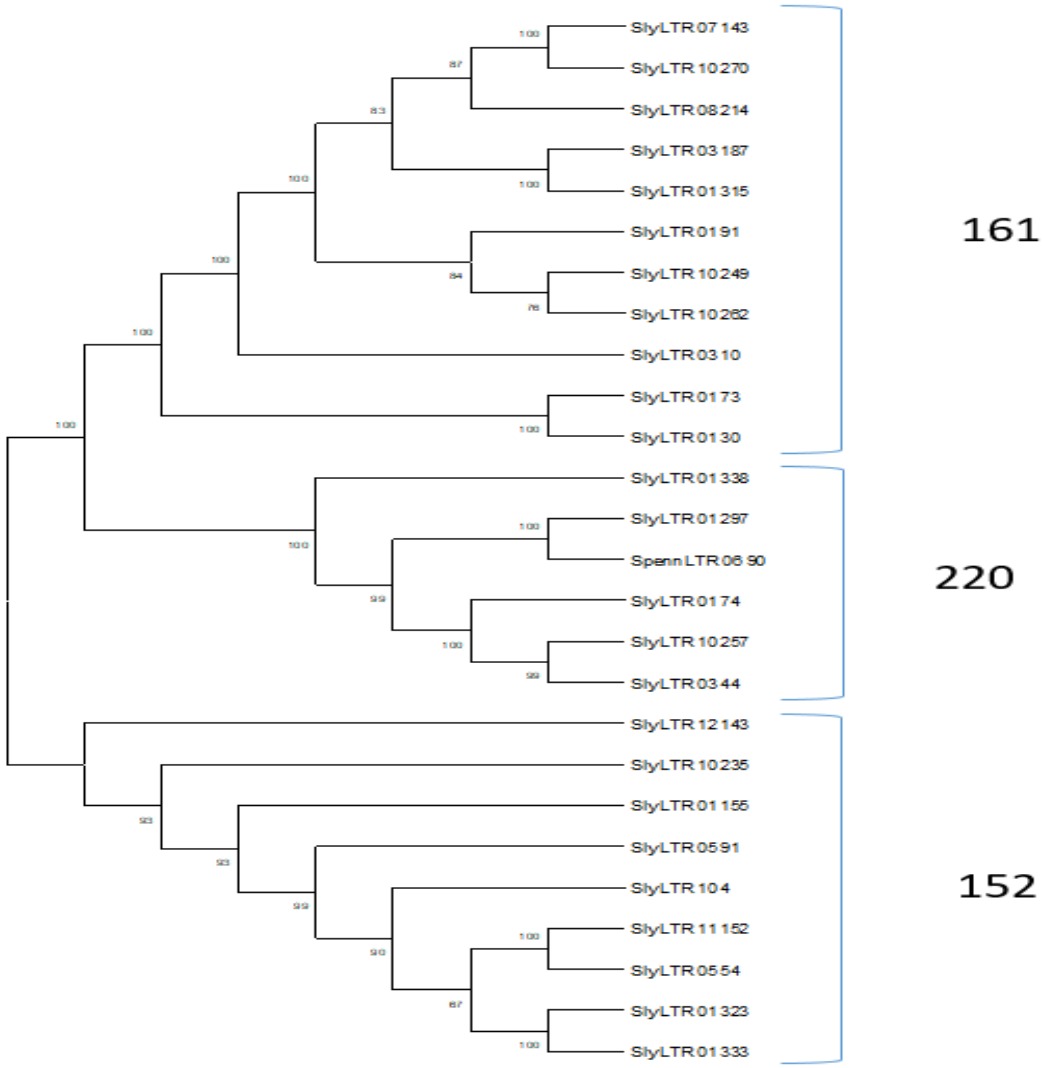
Figura 3: Expresión estimada retrotransposones LTR

En la figura se muestra la expresión estimada de retrotransposones LTR, inferida del análisis con Salmon. Los diagramas están divididos por *clusters*, de acuerdo al análisis derivado de Vsearch. Las barras muestran el promedio de las 2 repeticiones (en TPM). La barra muestra el promedio y la barra de error se utilizó para graficar los valores extremos. Por cada transposón se grafican 3 barras, cada una perteneciente a un tejido distinto: hoja (rojo), flor (verde) y meristema (azul). La escala del eje está en graduación logarítmica. Se muestra una línea en el punto de corte de 1TPM (rango de aceptación), cuando la totalidad de la barra de error cruza la línea de aceptación se toma ETs activo.

Al haber encontrado estos grupos, tal como prevé la literatura, nos da confianza el proceso de depuración realizado hasta ahora. Siguiendo por este camino, exploramos los datos disponibles para buscar familias de retrotransposones LTR novedosas cuyo escape no haya sido aún reportado. Mediante la metodología aplicada para los transposones MESSI y Rider, se encontraron otros 6 *clusters* con elementos activos (superando el valor *cut-off* 1TPM de nivel de transcripto estimado) y de similares características de expresión. Entre estos están los *clusters* 20, 53, 152, 216, 245 y 279, en los cuales no se encontró un resultado aceptable en el BLAST con MESSI o Rider, demostrando que no pertenecen a estas familias (Figura 3b).

Adicionalmente, para comprobar que los *clusters* novedosos no pertenecían a las familias ya mencionadas, se realizó una comparación filogenética con Rider y MESSI, cuidando de considerar que perteneciesen a las mismas superfamilias; es decir, Rider con el/los nuevos *clusters copia* y MESSI con el/los nuevos *clusters gypsy* (reconocidos por BLAST contra una base de datos pública: Gypsy Database; ver Llorens et al., (2010). Observamos que los elementos de escape novedosos pertenecían a familias diferentes pero filogenéticamente relacionadas a aquellas ya descriptas (Figura 4).

(a)


SlyLTR07143
SlyLTR10270
SlyLTR08214
SlyLTR03187
SlyLTR01315
SlyLTR0191
SlyLTR10249
SlyLTR10262
SlyLTR0310
SlyLTR0173
SlyLTR0130
SlyLTR01338
SlyLTR01297
SpennLTR0690
SlyLTR0174
SlyLTR10257
SlyLTR0344
SlyLTR12143
SlyLTR10235
SlyLTR01155
SlyLTR0591
SlyLTR104
SlyLTR11152
SlyLTR0554
SlyLTR01323
SlyLTR01333
161
220
152

(b)

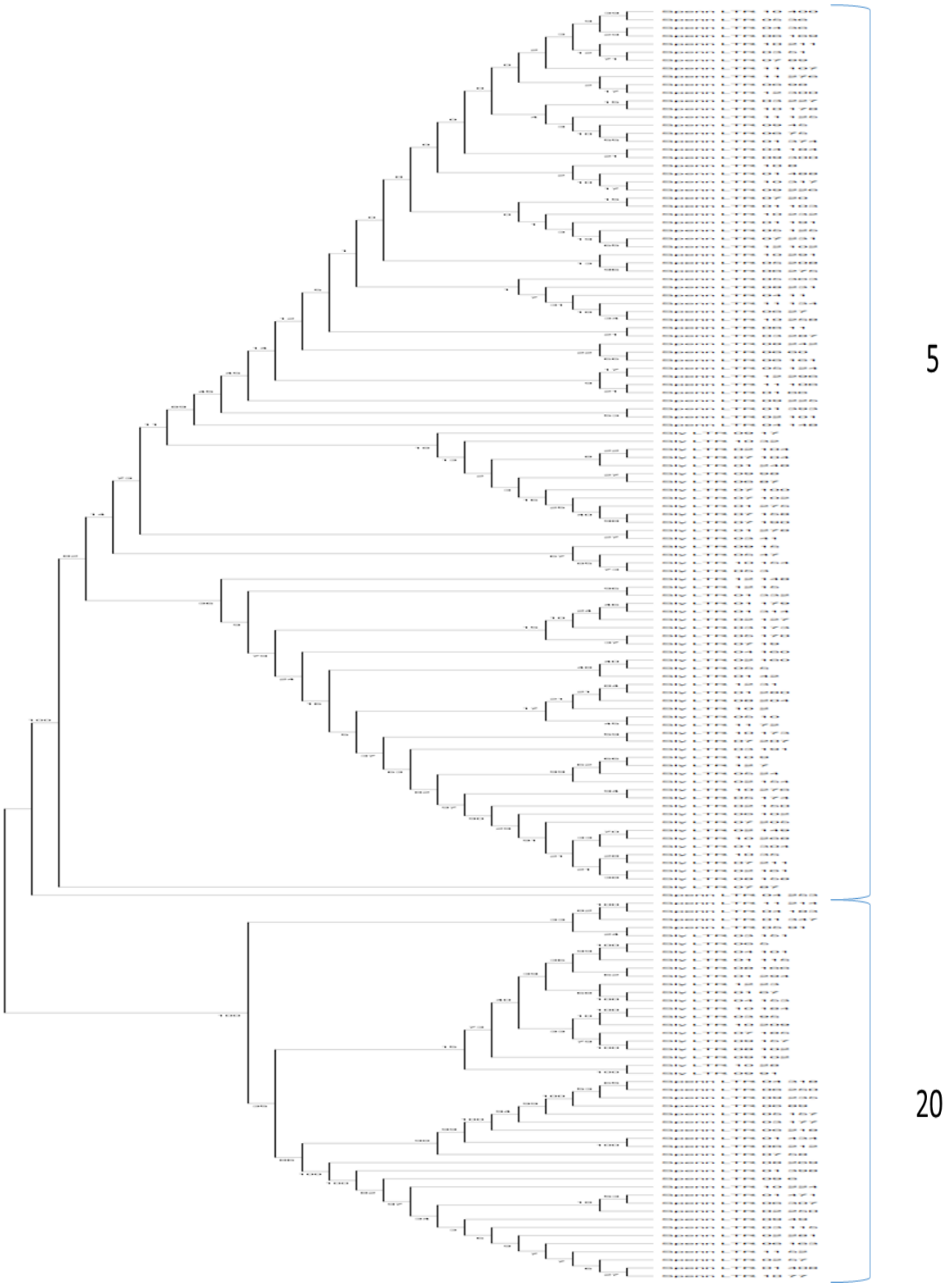
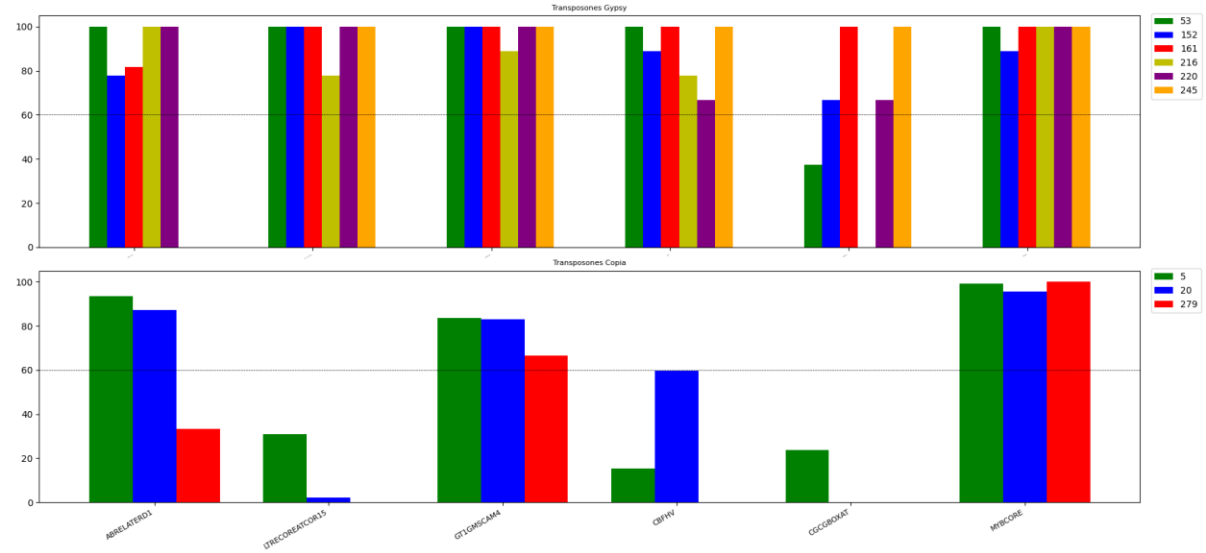


Figura 4: Árboles Filogenéticos

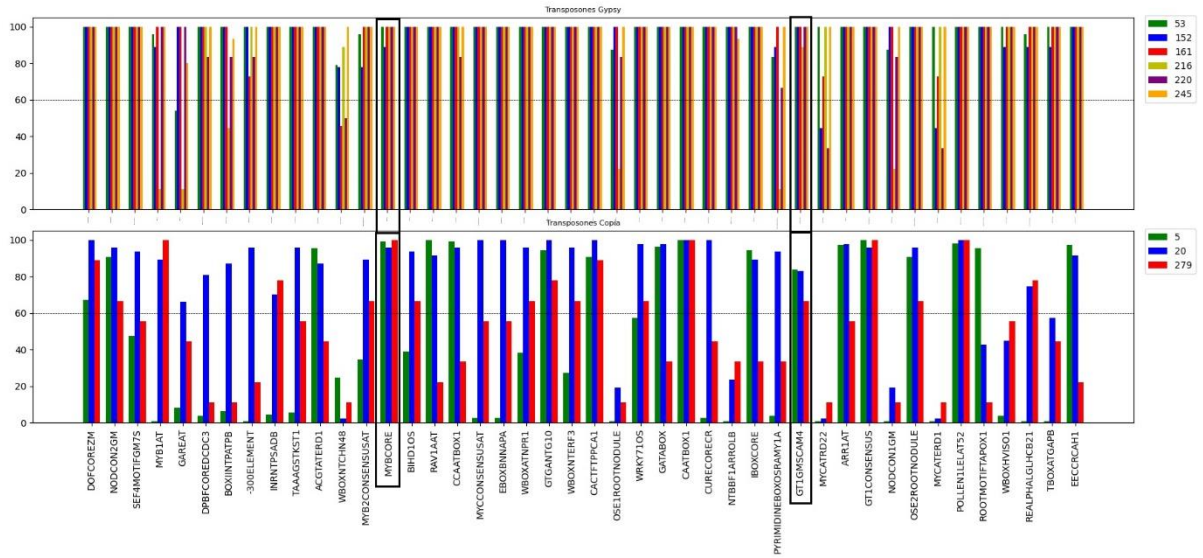
(a) Árbol ejemplificador ¿? con *clusters gypsy*, donde se puede apreciar la separación de los *cluster* y los transposones según lo esperado. Los *clusters* 161 y 220 son de la familia MESSI, mientras que el 152 no tuvo coincidencias con el original de MESSI y se consideraría una nueva familia de ETs. (b) Árbol ejemplificador ¿? con *clusters copia.*

Siendo que los *cis-elements* dentro de la zona promotora deben tener alguna participación en el escape, aunque no necesariamente sean el factor decisivo ni el único factor (Benoit et al., 2019), utilizamos *cis-elements* ya reconocidos (LTRECOREATCOR15, CGCGBOXAT, CBFHV, MYBCORE, ABRELATERD1, GT1GMSCAM4) con efecto en la expresión de Rider, realizándose un SIGNAL SCAN para encontrarlos en la plataforma PLACE. En efecto, estos *cis-elements* aparecieron en el *cluster* 5 (Figura 5a). Filtramos los elementos repetidos y se buscó la intersección entre los distintos *clusters.* Se encontraron que los tres (MYBCORE, ABRELATERD1, GT1GMSCAM4) *cis-elements* comunes dentro del *cluster* 5 de Rider eran también comunes a todos los *clusters copia* seleccionados, y solo dos (MYBCORE, GT1GMSCAM4) *cis-elements* fueron comunes a la intersección de ambas superfamilias *copia* y *gypsy*. Estos *cis-elements* superaban el 60% de aparición dentro de cada *cluster* (Figura 5b). Seguidamente se realizó un análisis similar con los *cis-elements* de MESSI relacionados con auxinas y meristema (Figura 5c), encontrando solo uno de ellos en absolutamente todos los *clusters* (esto es, *clusters copia+gypsy* específicos bajo análisis).

(a)


Transposones Gypsy
53
152
161
216
220
245
Transposones Copia
5
20
279
ABRELATERD1
LTRECOREATCOR15
GT1GMSCAM4
CBFHV
CGCGBOXAT
MYBCORE

(b)


Transposones Gypsy
53
152
161
216
220
245
Transposones Copia
5
20
279
DOFCOREZM
NODCON2GM
SEF4MOTIFGM7S
MYB1AT
GAREAT
DPBFCOREDCDC3
BOXIINTPATPB
-300ELEMENT
INRNTPSADB
TAAAGSTKST1
ACGTATERD1
WBOXNTCHN48
MYB2CONSENSUSAT
MYBCORE
BIHD1OS
RAV1AAT
CCAATBOX1
MYCCONSENSUSAT
EBOXBNNAPA
WBOXATNPR1
GTGANTG10
WBOXNTERF3
CACTFTPPCA1
OSE1ROOTNODULE
WRKY71OS
GATABOX
CAATBOX1
CURECORECR
NTBBF1ARROLB
IBOXCORE
PYRIMIDINEBOXOSPAMY1A
GT1GMSCAM4
MYCATRD22
ARR1AT
GT1CONSENSUS
NODCON1GM
OSE2ROOTNODULE
MYCATERD1
POLLEN1LELAT52
ROOTMOTIFTAPOX1
WBOXHVISO1
REALPHALGLHCB21
TBOXATGAPB
EECCRCAH1

(c)

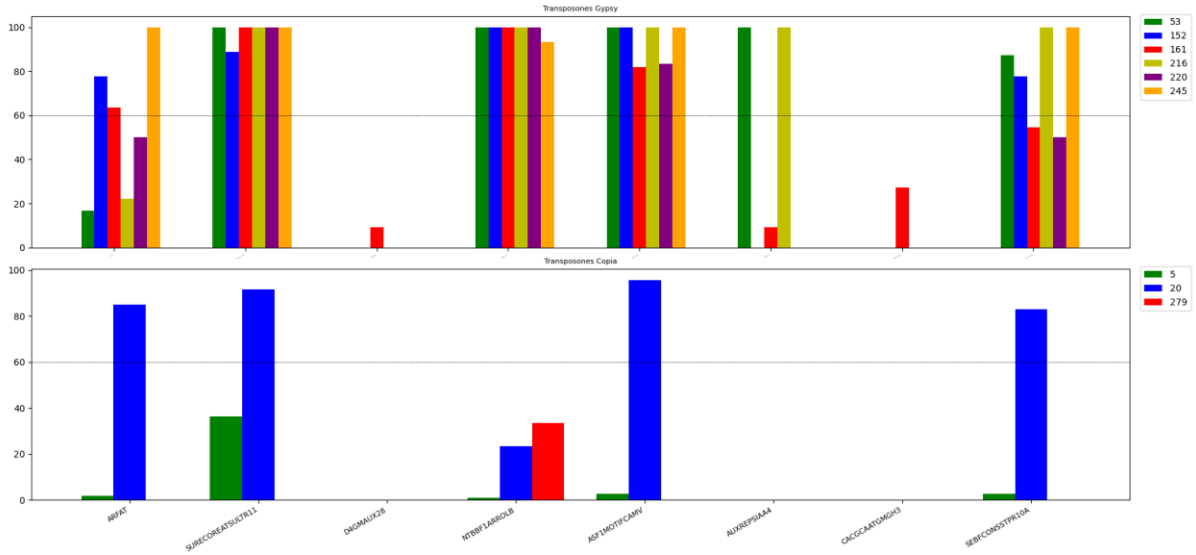


Figura 5: *cis-elements* en *clusters*

En esta figura se muestra la proporción de ocurrencia dentro de cada *cluster* (en porcentaje) de los *cis-elements* compartidos. Las figuras están divididas en dos sub-gráficos: el superior con *clusters* de la superfamilia *gypsy*, mientras que el inferior con los correspondientes *copia*. La figura (a) muestra *cis-elements* de Rider, la figura (b) muestra los elementos resultantes de la intersección de la dos superfamilias de ETs con las que trabajamos (los *cis-elements* en recuadros son los característicos de Rider) y la figura (c) los elementos relacionados con auxinas y meristema.

DISCUSIÓN

En este trabajo se propuso encontrar retrotransposones LTR en dos genomas disponibles de especies vegetales emparentadas filogenéticamente, mediante métodos de búsqueda y confirmación *in-silico* (es decir, métodos puramente informáticos). Esto implicó superar muchos desafíos de búsqueda y autenticación de los resultados, ya que no se pudieron confirmar de manera experimental.

El primer desafío a superar fue el de encontrar los elementos con sus correspondiente LTR, lo que en realidad resultó relativamente simple ya que existen softwares ya desarrollados para esta tarea. En este caso, se utilizó LTR Harvest (Ellinghaus et al., 2008). Este programa devolvió un sinnúmero de anotaciones

posibles, que debieron ser filtradas para evitar falsos positivos, y tomar secuencias similares a LTRs aunque tal vez no lo fueran. Estos falsos positivos pueden ser causados, por ejemplo, por secuencias duplicadas en tándem; dado que el *software* no verifica automáticamente lo que se encuentra en la secuencia INNER, puede suceder que dos secuencia muy similares sean confundidas con LTRs si se ajustan a los parámetros de búsqueda. Esto debió ser tenido en cuenta en todo el procesamiento de datos dada la gran cantidad de ETs, siendo imposible revisarlos uno por uno manualmente y teniendo que buscar alternativas para descartar aquellos falsos positivos.

Una vez convertido el *output* de LTR Harvest (Ellinghaus et al., 2008) a un archivo tipo BED y extraída la secuencia FASTA correspondiente, se realizó la separación en *clusters*. De esta forma, los ETs se agruparon por similitud de secuencia entre-LTRs, pudiéndose eliminar aquellos *clusters* que presentaban elementos únicos (falsos positivos, o retrotransposones LTR que no ocurren en forma de familias y que por lo tanto imposibilitan análisis comparativos). Con menor cantidad de secuencias, fue más simple el procesado informático y la construcción de árboles genealógicos que discutiremos más adelante.

Para despejar la incógnita de si los ETs “escapaban”, se utilizó el *sofware* Salmon (Patro et al., 2017) para estimar el nivel de transcripto de una secuencia, analizando ARN-seq disponibles. El software ajusta la salida por abundancia de *reads* no ambiguos, lo que es necesario debido a que la expresión de un ETs puede no solo ser baja, sino que también los *reads* potencialmente se alinean en más de un lugar en virtud de la cantidad de copias similares dentro de una familia de ETs, lo que causa errores en la cuantificación de la expresión. De esta manera, se obtuvo la Tabla 1 y la Figura 3, y se separaron aquellos elementos que superaban un *cut-off* de 1TPM. Se descartó hacer una comparación entre-tejidos no solo debido a las problemáticas de la estimación de expresión comparativa derivadas de la similitud entre-ETs, sino también porque se prefirió focalizarse en escapes asociados a la línea germinal donde existiría presión de selección evolutiva para escapar (a diferencia del soma, donde nuevas inserciones transposicionales son perdidas en la descendencia).

No se podría descartar que los transposones sean secundariamente amplificados debido a la activación transcripcional de un gen codificante cercano, y se buscó manualmente la posición cromosómica donde se encontraban ubicados los ETs, descartándose aquellos casos donde genes podrían estar amplificándolos (como se

ejemplifica en la Figura 2). Aunque los *clusters* que se obtuvieron se sabía que poseían ETs que escapaban, se logró encontrar *clusters* nuevos con ETs que posiblemente escapaban y no se halló literatura que los describan. Se concluye entonces, que el análisis *in-silico* permite encontrar *clusters* de transposones con elementos que escapan. Se procedió entonces a analizar las relaciones filogenéticas tras alineamientos de secuencia, no solo dentro de cada uno de estos *clusters* informativos, sino también entre *clusters* de superfamilias *gyspsy* y *copia*.

Al realizar éstos árboles genealógicos con los datos crudos, curiosamente daba una salida de árboles bimodales. Esto se debió a que no se ecualizó/estandarizó el sentido de la secuencia y los ET se pueden insertar de dos posibles formas en el sitio de unión siendo la misma secuencia como se explica en Drost & Sanchez (2019). Por lo tanto algunas se encontraban en *forward* y otras en *reverse* lo que al realizar el árbol generaba dos ramas principales. Se realizó el reverso-complementario de las secuencias que estaban en sentido opuesto, quedando todas en el mismo sentido. Se volvió a realizar los árboles y ésta vez resultaron apropiados.

Los ETs que se activan transposicionalmente mantienen jóvenes a sus correspondientes familias, ya que copian constantemente su secuencia y re-insertan copias modernas. Nótese, sin embargo, que estas nuevas copias no necesariamente van a ser exactamente iguales al elemento parental, ya que en el ciclo de vida pueden suceder cambios por recombinación o por acumulacion de errores durante la transcripción-reversa (Drost & Sanchez, 2019). Los árboles, tal como se esperaba, generaron sub-grupos de secuencias apilando juntas las transposiciones más modernas en virtud de su amplificación por ráfaga transposicional reciente; mientras que las secuencias más antiguas tienden a aparecer aisladas por desaparición de congéneres, lo que las ubica más cercanas a la base del árbol. Además, tienden a alejarse filogenéticamente debido a la acumulación de mutaciones por permanencia extendida en el genoma hospedador.

Se realizaron los árboles con las secuencias entre-LTRs (INNER) y 5LTR por separado en una comparativa de los resultados. Se espera que si en estas secuencias se observa una misma cantidad de cambios, el porcentaje de cambio total es diferente, debido a sus diferencias de tamaño. En estos árboles, aunque son un poco diferentes, los subgrupos de ET se mostraron similares.

El siguiente punto que se debería comprobar es cuáles de los elementos y *clusters* encontrados en este trabajo son ya conocidos, y si existe alguno nuevo. ETs

de escape ya descriptos en *S. lycopersicum* -Rider y MESSI tomados de los trabajos de Benoit et al. (2019) y Sanchez et al. (2019)- se usaron como controles positivos, realizándose con sus secuencias BLAST confirmatorios. Encontramos que el *cluster* 5 pertenecía a la familia Rider, y los *clusters* 161 y 220 a MESSI, con todas las características descriptas por los autores ya mencionados, confirmando el éxito de nuestro procedimiento exploratorio. También mediante este análisis se pudo determinar a qué familia pertenecía cada *cluster* y hacer nuevamente los árboles (dando por ejemplo la Figura 4, donde se separan todos los ETs exactamente como se esperaba). Ergo, podemos decir con una elevada certeza que los otros *clusters* con ET todavía no descriptos probablemente reflejen *clusters* de escape. En este punto, si se quisiese profundizar más en los niveles estimados de transcriptos o bien ir más allá de la exploración informática que es puramente de secuencia conocida, se tendrían que realizar experimentos de validación en laboratorio. Esto no estaba dentro del proyecto del trabajo.

En el análisis de las secuencias, se encontraron algunos resultados inesperados como un transposón muy pequeño (en los términos de los transposones LTR típicamente descriptos) que aparentemente escapaba al silenciamiento dentro de una familia pequeña de transposones cortos.

Finalmente, se realizó una búsqueda de *cis-elements* con un doble propósito: el primero y principal fue hacer una comprobación independiente sobre los *cis-elements* encontrados, verificando si cumplen las características descriptas anteriormente por otros grupos de investigación (Benoit et al., 2019); y en segundo lugar, fue buscar algún patrón que dé una evidencia de por qué éstos ETs escapan. Si estos *cis-elements* estuviesen implicados en la transcripción tejido-específica de algún transposón, evolutivamente se esperaría que aparezca con más frecuencia que otros. Mediante la plataforma PLACE (Higo et al., 1999) se buscó todos los *cis-elements* del *cluster* de Rider, se descartaron las repeticiones, y se vio aproximadamente en qué proporción relativa aparecían. Como era esperable, se encontraron cada uno de los *cis-elements* ya descriptos en Rider por la literatura; aunque en algunos casos en muy bajas proporciones dentro del *cluster* sugiriendo que no podrían estar involucrados en el proceso de escape.

Continuando con la exploración de los *cis-elements,* se buscó cuáles aparecían en todos los ETs de *clusters copia* de escape; curiosamente, aparecieron 2 (MYBCORE, GT1GMSCAM4) *cis-elements* de Rider en la totalidad de los *clusters*

evaluados, registrándose en más del 60% de los ETs individuales dentro de cada grupo. Esto incrementa las chances de que no sea casual su ocurrencia. Se intentó hacer algo similar con los *clusters* de MESSI y relacionados, pero al no existir *cis-elements* descriptos para estos, se exploraron secuencias relacionadas con auxinas y meristemas (lugar anatómico donde típicamente escapa MESSI). En este caso, solo tres *cis-elements* se encontraron en todos los *clusters gypsy* estudiados, y al menos uno de estos estaba presente en la totalidad de los *clusters*.

Analizando los resultados, la proporción de los *cis-elements* compartidos en todos los *clusters* no parece formar un patrón obvio de aparición (según las familias Rider o MESSI; o bien correlacionado con los *clusters* de *gypsy* o *copia*). Sin embargo, parece haber una correlación clara en la aparición de *cis-elements* con el tamaño del transposón; es decir, que cuanto más larga es la secuencia del ETs, más probable es que aparezca un *cis*-elemento determinado. Esto no quiere decir que este tipo de secuencias no estén implicadas en el escape, pero sí que no existe un tipo de secuencia determinante para el escape. Para demostrar fehacientemente la implicancia en el fenómeno de escape se deberían hacer pruebas experimentales adicionales.

Como conclusión final, demostramos que efectivamente se pueden buscar y encontrar ETs por medio de análisis *in-silico* de genomas secuenciados con un razonable nivel de confiabilidad y con buena posibilidad de encontrar *clusters* de escape novedosos. Este proceso también puede ser extrapolado a otros organismos, probablemente sin demasiados cambios en la estrategia. Lo que se pudo realizar en este trabajo fue intentar profundizar en las causas que expliquen por qué estas secuencias “egoístas” escapan al silenciamiento, y nuestra estrategia de procesamiento de ETs podría ser extendida para generar un algoritmo de búsqueda de elementos escapistas informativos. Sin embargo, nótese que por el tipo de información genómica disponible, nos fue posible analizar comparativamente solo una parte de la historia –el escape al silenciamiento transcripcional (TGS). Este es solo uno de los tantos *check-points* donde los ETs tendrían la posibilidad de escapar al silenciamiento del hospedador; otro análisis factible en el futuro podría ser el estudio al escape post-transcripcional (PTGS).

BIBLOGRAFIA

Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ (1990) Basic local alignment search tool. J Mol Biol **215:** 403-410

Anders S, Pyl PT, Huber W (2015) HTSeq--a Python framework to work with high-throughput sequencing data. Bioinformatics **31:** 166-169

Benoit, M., Drost, H.-G., Catoni, M., Gouil, Q., Lopez-Gomollon, S., Baulcombe, D., and Paszkowski, J. (2019). Environmental and epigenetic regulation of Rider retrotransposons in tomato. PLOS Genetics, 15(9), e1008370. https://doi.org/10.1371/journal.pgen.1008370

Bolger A, Scossa F, Bolger ME, Lanz C, Maumus F, Tohge T, Quesneville H, Alseekh S, Sørensen I, Lichtenstein G, Fich EA, Conte M, Keller H, Schneeberger K, Schwacke R, Ofner I, Vrebalov J, Xu Y, Osorio S, Aflitos SA, Schijlen E, Jiménez-Goméz JM, Ryngajllo M, Kimura S, Kumar R, Koenig D, Headland LR, Maloof JN, Sinha N, van Ham RCHJ, Lankhorst RK, Mao L, Vogel A, Arsova B, Panstruga R, Fei Z, Rose JKC, Zamir D, Carrari F, Giovannoni JJ, Weigel D, Usadel B, Fernie AR (2014) The genome of the stress-tolerant wild tomato species Solanum pennellii. Nature Genetics **46:** 1034

Bolger AM, Lohse M, Usadel B (2014) Trimmomatic: a flexible trimmer for Illumina sequence data. Bioinformatics **30:** 2114-2120

Bousios A, Gaut BS (2016) Mechanistic and evolutionary questions about epigenetic conflicts between transposable elements and their plant hosts. Curr Opin Plant Biol **30:** 123-133

Cerutti H, Casas-Mollano JA (2006) On the origin and functions of RNA-mediated silencing: from protists to man. Curr Genet **50:** 81-99

Consortium TG (2012) The tomato genome sequence provides insights into fleshy fruit evolution. Nature **485:** 635-641

Devos K. M. *et al.* (2002) Genome Size Reduction through Illegitimate Recombination Counteracts Genome Expansion in Arabidopsis.

Dobin A, Davis CA, Schlesinger F, Drenkow J, Zaleski C, Jha S, Batut P, Chaisson M, Gingeras TR (2013) STAR: ultrafast universal RNA-seq aligner. Bioinformatics **29:** 15-21

Drost, H.-G., and Sanchez, D. H. (2019). Becoming a Selfish Clan: Recombination Associated to Reverse-Transcription in LTR Retrotransposons. Genome Biology and Evolution, 11(12), 3382-3392. https://doi.org/10.1093/gbe/evz255

Edgar RC (2004) MUSCLE: multiple sequence alignment with high accuracy and high throughput. Nucleic Acids Research **32:** 1792-1797

Ellinghaus D, Kurtz S, Willhoeft U (2008) LTRharvest, an efficient and flexible software for de novo detection of LTR retrotransposons. BMC Bioinformatics **9:** 18

Fultz D, Choudury SG, Slotkin RK (2015) Silencing of active transposable elements in plants. Curr Opin Plant Biol **27:** 67-76

Galindo-González, L., Mhiri, C., Deyholos, M. K., and Grandbastien, M.-A. (2017). LTR-retrotransposons in plants: Engines of evolution. Gene, 626, 14-25. https://doi.org/10.1016/j.gene.2017.04.051

Grandbastien MA (2015) LTR retrotransposons, handy hitchhikers of plant regulation and stress response. Biochim Biophys Acta **1849:** 403-416

Higo, K., Ugawa, Y., Iwamoto, M., and Korenaga, T. (1999). Plant cis-acting regulatory DNA elements (PLACE) database: 1999. Nucleic Acids Research, 27(1), 297-300. https://doi.org/10.1093/nar/27.1.297

Jiang N, Gao D, Xiao H, van der Knaap E (2009) Genome organization of the tomato sun locus and characterization of the unusual retrotransposon Rider. Plant J **60:** 181-193

Jouffroy O, Saha S, Mueller L, Quesneville H, Maumus F (2016) Comprehensive repeatome annotation reveals strong potential impact of repetitive elements on tomato ripening. BMC Genomics **17:** 624

Krueger F, Andrews SR (2011) Bismark: a flexible aligner and methylation caller for Bisulfite-Seq applications. Bioinformatics **27:** 1571-1572

Kumar, S., Stecher, G., Li, M., Knyaz, C., & Tamura, K. (2018). MEGA X: Molecular Evolutionary Genetics Analysis across Computing Platforms. Molecular Biology and Evolution, 35(6), 1547–1549. https://doi.org/10.1093/molbev/msy096

Langmead B, Salzberg SL (2012) Fast gapped-read alignment with Bowtie 2. Nat Methods **9:** 357-359

Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, Durbin R (2009) The Sequence Alignment/Map format and SAMtools. Bioinformatics **25:** 2078-2079

Llorens, C., Futami, R., Covelli, L., Dominguez-Escriba, L., Viu, J. M., Tamarit, D., Aguilar-Rodriguez, J., Vicente-Ripolles, M., Fuster, G., Bernet, G. P., Maumus, F., Munoz-Pomer, A., Sempere, J. M., Latorre, A., and Moya, A. (2010). The Gypsy Database (GyDB) of mobile genetic elements: release 2.0. Nucleic Acids Research, 39(Database), D70-D74.

Lodish H, Berk A, Matsudaira P, Kaiser CA, Krieger M, Scott MP, Zipursky L, Darnell J (2005) Biologia celular y molecular 5 ed. Editorial Medica Panamericana 414p

Ma J, Devos KM, Bennetzen JL (2004) Analyses of LTR-retrotransposon structures reveal recent and rapid genomic DNA loss in rice. Genome Res **14:** 860-869

Maumus, F., and Quesneville, H. (2016). Impact and insights from ancient repetitive elements in plant genomes. Current Opinion in Plant Biology, 30, 41–46. https://doi.org/10.1016/j.pbi.2016.01.003

Ou S, Jiang N (2018) LTR_retriever: A Highly Accurate and Sensitive Program for Identification of Long Terminal Repeat Retrotransposons. Plant Physiol **176:** 1410-1422

Park SJ, Jiang K, Schatz MC, Lippman ZB (2012) Rate of meristem maturation determines inflorescence architecture in tomato. Proc Natl Acad Sci U S A **109:** 639-644

Patro, R., Duggal, G., Love, M. I., Irizarry, R. A., & Kingsford, C. (2017). Salmon provides fast and bias-aware quantification of transcript expression. Nature Methods, 14(4), 417–419. https://doi.org/10.1038/nmeth.4197

Pierce BA. (2006) Genetica: un enfoque conseptual 2ed. Editorial Medica Panamericana 300p

Quinlan AR, Hall IM (2010) BEDTools: a flexible suite of utilities for comparing genomic features. Bioinformatics **26:** 841-842

Robinson MD, McCarthy DJ, Smyth GK (2010) edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. Bioinformatics **26:** 139-140

Rognes T, Flouri T, Nichols B, Quince C, Mahé F (2016) VSEARCH: a versatile open source tool for metagenomics. PeerJ **4:** e2584

Sanchez, D. H., Gaubert, H., & Yang, W. (2019). Evidence of developmental escape from transcriptional gene silencing in MESSI retrotransposons. New Phytologist, 223(2), 950–964.

Stamatakis A (2014) RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. Bioinformatics **30:** 1312-1313

Tomato Genome Sequencing C, Aflitos S, Schijlen E, de Jong H, de Ridder D, Smit S, Finkers R, Wang J, Zhang G, Li N, Mao L, Bakker F, Dirks R, Breit T, Gravendeel B, Huits H, Struss D, Swanson-Wagner R, van Leeuwen H, van Ham RC, Fito L, Guignier L, Sevilla M, Ellul P, Ganko E, Kapur A, Reclus E, de Geus B, van de Geest H, Te Lintel Hekkert B, van Haarst J, Smits L, Koops A, Sanchez-Perez G, van Heusden AW, Visser R, Quan Z, Min J, Liao L, Wang X, Wang G, Yue Z, Yang X, Xu N, Schranz E, Smets E, Vos R, Rauwerda J, Ursem R, Schuit C, Kerns M, van den Berg J, Vriezen W, Janssen A, Datema E, Jahrman T, Moquet F, Bonnet J, Peters S (2014) Exploring genetic variation in the tomato (Solanum section Lycopersicon) clade by whole-genome sequencing. Plant J **80:** 136-148

Wicker T. ***et al.*** (2007) A unified classification system for eukaryotic transposable elements.

Xiao H, Jiang N, Schaffner E, Stockinger EJ, van der Knaap E (2008) A Retrotransposon-Mediated Gene Duplication Underlies Morphological Variation of Tomato Fruit. Science **319:** 1527-1530

Xu Y, Du J (2014) Young but not relatively old retrotransposons are preferentially located in gene-rich euchromatic regions in tomato (Solanum lycopersicum) plants. Plant J **80:** 582-591

ANEXO I

SOFTWARE Y FORMATOS

- Línea de comando de Linux
- LTR Harvest (Ellinghaus et al., 2008)
- PYTHON en Linux
- Vsearch clustering (Rognes et al., 2016)
- Bedtools getfasta (Quinlan & Hall, 2010)
- Muscle maketree (Edgar, 2004)
- Excel
- Power point
- MEGA Software (Kumar et al., 2018)
- Salmon (Patro et al., 2017)
- BLAST
- Seqmonk
- Formato FASTA
- Formato BED
- Formato GFF3

LTR Harvest

LTR Harvest (Ellinghaus et al., 2008) es una herramienta de software para predicción *de novo* de retrotransposon LTR. El software calcula las posiciones límites de los posibles retrotrasposones de LTR en una estructura d índice de la secuencia objetivo genómico. La matriz de los sufijos mejorada es el genoma del cual se quiere extraer los ET. En base a la estructura de los ET, el software busca estructuras genómicas que posiblemente pertenezcan a un elemento de este tipo. LTR Harvest (Ellinghaus et al., 2008) implementa varios filtros, aplicándose consecutivamente en la secuencia de datos para rechazar candidatos que no cumplan con la característica de la secuencia, longitud o distancia del ET. Eligiendo los filtros según la familia de ETs que queremos encontrar.

PYTHON

Es un lenguaje de programación que hace hincapié en la legibilidad de su código. Es un lenguaje de programación con una forma de escritura (sintaxis) de fácil lectura. Es lo que se conoce como un lenguaje de *scripting*, que puede ser ejecutado por partes y no necesitan ser compilado en un paso aparte. Es un lenguaje sencillo de aprender y programar sin estudios específicos de programación.

VSearch

Vsearch (Rognes et al., 2016) es una herramienta que fue utilizada para hacer los *clusters* de las secuencias, de código abierto y gratuito para procesar y preparar datos de secuencias de nuclétidos según su homología; para aplicaciones de genómica, poblaciones, etc. Utiliza en primer lugar una serie de palabras llamados *k-mers*, que son fragmentos de las secuencias de ADN recurrentes en todas secuencias que se desean alinear, y se crea una base de datos con los *k-mer* únicos y su posición en la secuencia. La segunda parte de la búsqueda se inicia con el ordenamiento según cuántos *k-mer* únicos contenga una secuencia; ésta será la semilla inicial y el resto se alineará en base a ésa, mediante una alineación global. Por último, se puntuará según el valor especificado, aceptando o rechazando el alineamiento.

Me parece tambien impartante aclarar que efeccto tine cada parametro para un mejor entendimiento del *output* del *software.*

--*threads* 4: Número de subprocesos de cómputo a usar (1 a 256). El número de subprocesos debe ser menor o igual al número de núcleos de CPU disponibles

--*cluster_smallmem*: Agrupa las secuencias de FASTA en el nombre de archivo sin modificar automáticamente su orden de antemano. Se espera que la secuencia se clasifique disminuyendo la longitud de la secuencia, a menos que se use --*usersort*.

--*usersort*: Cuando use --*cluster_smallmem*, permita cualquier orden de entrada de secuencia, no solo un orden de longitud decreciente

--*id* 0.8: Rechaza la coincidencia de la secuencia si la identidad apareamiento es inferior real (valor que oscila de 0,0 a 1,0 incluido).

--*iddef* 4: Definición de BLAST, equivalente a --*iddef* 1 (distancia de edición: (columnas coincidentes) / (longitud de alineación))para alineaciones globales por pares.

--*maxaccepts* 0: Número máximo de aciertos para aceptar antes de detener la búsqueda. El valor predeterminado es 1. Esta opción funciona a la par con --*maxrejects*

--*maxrejects* 0: Número máximo de secuencias de destino no coincidentes a considerar antes de detener la búsqueda. El valor predeterminado es 32. Esta opción funciona en par con --*maxaccepts*. --> Si --*maxaccepts* y --*maxrejects* se establecen en 0, se busca en la base de datos completa.

--*strand both*: Busca secuencias similares, marca solo la cadena positiva (por defecto) o marque ambas cadenas.

--*uc*: La búsqueda de resultados da como resultado un nombre de archivo utilizando un formato similar a *uclust* separado por tabulaciones con 10 columnas.

Bedtools

Es una herramienta con múltiples funciones para los análisis genómicos comunes, permite el análisis aritmético de las secuencias como por ejemplo fusionar, cortar, completar y barajar entre varias secuencias. Permite trabajar con formatos BEM, BED, GFF/GTF, VCF. El programa permite hacer la tarea de manera simple o también se puede hacer análisis más complejos utilizando varias herramientas en simultáneo. En nuestro caso, vamos a utilizar mayormente la herramienta para poder extraer las secuencias de nucleótidos y tomar de la posición de los ETs, la secuencia de nucleótidos de los archivos de genomas.

MUSCLE

MUSCLE (Edgar, 2004) es un programa para crear múltiples alineamientos de secuencias. Los elementos del algoritmo incluyen la estimación de distancia rápida usando el conteo de *k-mer*, la alineación progresiva usando una función llamada puntaje de expectativa de registro y el refinamiento usando la partición restringida dependiente del árbol. La velocidad y la precisión de MUSCLE (Edgar, 2004) es comparable a otros *software* de alineamiento, siendo más rápido y con menos requerimientos computacionales.

MEGA *software*

Es un *software* para realizar análisis estadístico de la evolución molecular de ADN Y proteínas, fue diseñado con una interfaz gráfica para un uso más simple, como para el análisis y visualización de las secuencias y los árboles producidos a partir de los alineamientos; dándonos una herramienta poderosa para el análisis científico de los *clusters*.

Salmon

Salmon (Patro et al., 2017) es un software para estimar, de manera rápida y precisa, el nivel del transcripto de un gen a partir de datos de ARN-seq. Este programa además de poder mapear, también permite cuantificar con enorme precisión la expresión de una secuencia. Esto lo logra mediante algoritmos que utilizan sesgos específicos de cada muestra, como los sesgos de GCs, sesgos que tienen en cuenta la posición para la cobertura de una región, los sesgos de secuenciación en los extremos 5' y 3' de los fragmentos, la distribución de fragmentos de distintas longitudes e incluso el uso de métodos específicos de la cadena. El algoritmo del salmon (Patro et al., 2017) es capaz de aprender de esos sesgos específicos de la muestra y utilizarlos para estimar la abundancia de los *read* en la sección y cuantificar la expresión.

BLAST

La herramienta de alineación local (BLAST) encuentra regiones de similitud local entre secuencias. El programa compara las secuencias de nucleótidos o proteínas con bases de todas las secuencias y calcula la importancia estadística de las coincidencias. El algoritmo se inicia con una aproximación de alineación que se optimiza a una medida de similitud local, esto es la puntuación del par de segmentos máximos (MSP). Los resultados matemáticos recientes sobre las propiedades estocas de los puntajes de MSP permiten un análisis del rendimiento de este método, así como la importancia estadística de las alineaciones que genera.

Seqmonk

Este *software* es un programa que permite visualizar y analizar un gran conjunto de regiones genómicas en modo de *genome-browser*. Es un programa que permite trabajar con datos de secuenciación de alto rendimiento, pero también

permite trabajar con cualquier conjunto de lecturas mapeadas. Seqmonk permite la navegación sobre el genoma anotado de forma rápida y visual, también incluye un conjunto de herramientas que le permiten cuantificar y filtrar los datos para encontrar regiones de interés en forma sistemática.

Formato FASTA

El formato FASTA es un formato de secuencia de nucleótidos o amino-ácidos. Es un formato que comienza con una descripción de una sola línea, seguida de líneas de datos de secuencia. La línea de definición se distingue de los datos de secuencia por un símbolo de mayor (>) al principio. La palabra que sigue es el identificador de la secuencias, además, unas descripciones opcionales separadas por barras. El identificador dependerá del nombre que se dé a la secuencia. La línea que sigue por debajo de la descripción tendrá representada la secuencia de nucleótidos o aminoácidos, las cuales estarán representadas en los códigos estándar IUB/IUPAC.

Formato GFF3

Este formato estándar para almacenar características genómicas en un formato de texto. Las características se separan en nueve columnas por tabulador. Las columnas poseen el siguiente formato: la primera es el ID de la secuencia, la segunda es el software fuente, la tercera es tipo, la cuarta y quinta columna son el inicio y fin de la secuencia, le sigue en la 6° posición el puntaje, continuado por el filamento en la séptima posición, anteúltimo está la fase y al final se le puede agregar atributos adicionales.

Formato BED

A diferencia del formato GFF3, este formato solo tiene 5 columnas: 1: cromosoma, 2 y 3: inicio y final, 4: tag de la secuencia y finalmente datos adicionales. Además, con el *software bedtool*, las secuencias se pueden manipular más fácilmente y con menor requerimiento de computacionales.

PARAMETROS

LTR Harvest

-v →*verbose mode*

-mintsd 3 → largo mínimo de los *motivf*

-maxtsd 20 → largo máximo de los *motivf*

-seed 30 →longitud de semilla mínima para repeticiones máximas exactas

-xdrop 5 →valores para la máxima extensión de semilla

-mat 2 →*matchscore*

-mis -2 →*mismatchscore*

-ins -3 →*insertionscore*

-del -3 →*deletionscore*

-minlenltr →100 longitud mínima de los LTR

-maxlenltr →7000 longitud máxima de los LTR

-mindistltr →1000 distancia mínima entre LTRs

-maxdistltr →30000 distancia máxima entre LTRs

-similar 97 → *Similarity* entre los LTRs

-overlaps best → en caso de anidados informa el ET con mejor similaridad de sus LTRS

-vic 60 → números de nucleótidos donde se buscarán los motivos terminales

Vsearch funcion clustering

vsearch randomize-

$ vsearch --threads 4 --shuffle 5LTR_Allnoncanonical_plus_unique_FINAL_LTR Harvest_together.fasta --randseed 1000 --output Random_5LTR_Allnoncanonical_plus_unique_FINAL_LTRharvest_together.fasta &

$ vsearch --threads 4 --shuffle INNER_Allnoncanonical_plus_unique_FINAL_LTRharvest_together.fasta --randseed 1000 --output Random_INNER_Allnoncanonical_plus_unique_FINAL_LTRharvest_together.fasta &

vsearch clustering

$ vsearch --threads 4 --cluster_smallmem Random_5LTR_Allnoncanonical_plus_unique_FINAL_LTRharvest_together.fasta --usersort --id 0.8 --iddef 4 --maxaccepts 0 --maxrejects 0 --strand both --uc 5LTR_toghether_output_vsearch_id08.uc &

$ vsearch --threads 4 --cluster_smallmem Random_5LTR_Allnoncanonical_plus_unique_FINAL_LTRharvest_together.fasta --usersort --id 0.9 --iddef 4 --maxaccepts 0 --maxrejects 0 --strand both --uc 5LTR_toghether_output_vsearch_id09.uc &

$ vsearch --threads 4 --cluster_smallmem Random_INNER_Allnoncanonical_plus_unique_FINAL_LTRharvest_together.fasta --usersort --id 0.9 --iddef 4 --maxaccepts 0 --maxrejects 0 --strand both --uc INNER_toghether_output_vsearch_id08.uc &

$ vsearch --threads 4 --cluster_smallmem Random_INNER_Allnoncanonical_plus_unique_FINAL_LTRharvest_together.fasta --usersort --id 0.9 --iddef 4 --maxaccepts 0 --maxrejects 0 --strand both --uc INNER_toghether_output_vsearch_id09.uc &

MUSCLE

$ muscle -in seq.fasta -clwout seq.aln -tree2 tree_seq.phy -cluster neighborjoining

MEGA

- *Neigthbor-joining*
- *Bootstraps*: 1000
- *model*: *No. of diferences: gamma distributed with invariant sites*
- *gamma parameter*: 0.05
- *gaps: complete deletion*

Vsearch funcion *usearch*

$ vsearch --usearch_global *query*.fasta --db *cluster*.fasta --id 0.8 --uc output.uc

ANEXO II

SCRIPTS PYTHON

>Gff3 to bed

```
import time
start_time=time.time()
import subprocess

### to run in command line ###

### path ###
path = "/home/agustin/Documentos/LTR_retroTE/pruebas/"

### modificador ###

chromosomes_Dict = {0:"00",1:"01",2:"02",3:"03",4:"04",5:"05",6:"06",7:"07",8:"08",9:"09",10:"10",11:"11",12:"12"}

out_file_bed = open ("%sallnoncanonical_plus_unique_FINAL_LTRharvest_S_lycopersicum_chromosomes.bed" %(path), 'w' )

inputFile = '%sAllnoncanonical_plus_unique_FINAL_LTRharvest_S_lycopersicum_chromosomes.3.00_MODIF_sorted.gff3' %(path)

with open (inputFile) as infile:
    for in_line in infile:

        if in_line.startswith('#'):continue
        else:
```

```
        line2 = [x for x in in_line.strip('\n').split(" ") if x !=""]
        a,b,c,d,e,f,g,h,j,k,l,ll,m,o,p                                                    =
line2[0],line2[1],line2[2],line2[3],line2[4],line2[5],line2[6],line2[7],line2[8],line2[9],line2[
10],line2[11],line2[12],line2[13],line2[14]
        recover_value = chromosomes_Dict.get(int(p))
        Chr = "SL3.0ch"+str(recover_value)



        start_ = str(a)
        end_ = str(b)                          #get the start and end
        tag= Chr+":"+str(start_)+"-"+str(end_)      # generate name

        if start_=='0': start_='1'              # to avoid error: 'start too small' from HTSeq

        source_='LTRharvest'
        type_='exon'                              #for the column 3 (type) for exon, to call
expression
        score_='0'
        strand_='+'
        phase_='.'

        attributes = []     #list to generates the attributes
        attributes.insert (0,'Parent=%s' % tag)      # modify the attributes adding name
        attributes.insert (1,'LTRsimilarity=%s' %o)
        attributes.insert (2,'motive=Unk')
        attributes.insert (3,'Chr=%s' %Chr)
        attributes.insert (4,'start(TE)=%s' %a)
        attributes.insert (5,'end(TE)=%s' %b)
        attributes.insert (6,'size(TE)=%s' %c)
        attributes.insert (7,'s(5LTR)=%s' %d)
        attributes.insert (8,'e(5LTR)=%s' %e)
        attributes.insert (9,'size(5LTR)=%s' %f)
        attributes.insert (10,'s(3LTR)=%s' %j)
```

```
        attributes.insert (11,'e(3LTR)=%s' %k)
        attributes.insert (12,'size(3LTR)=%s' %l)
        attributes.insert (13,'TSD=%s' %ll)
        attributes.insert (14,'size_TSD=%s' %m)
        attributes_ = ";".join(attributes)          # reconstitute the attributes_ the new attributes name


        #write .bed4
        out_file_bed.write(Chr+"\t"+start_+"\t"+end_+"\t"+tag+"\n")

############################################################
###scrip resuelto para tranformar a .bed archivos .gff3
############################################################

import time
start_time = time.time()
import subprocess

### to run in command line ###

### path ###
path = "/home/agustin/Documentos/LTR_retroTE/pruebas/"

### modificador ###



out_file_bed = open ("%sallnoncanonical_plus_unique_FINAL_LTRharvest_S_lycopersicum_chromosomes.bed" %(path), 'w' )
```

```
inputFile = '%sAllnoncanonical_plus_unique_FINAL_LTRharvest_S_lycopersicum_chromosomes.3.00_MODIF_sorted.gff3' %(path)

with open (inputFile) as infile:

    for in_line in infile:

        if in_line.startswith('#'):continue
        else:
            line2 =[x for x in in_line.strip('\n').split('\t') if x !=""]
            #print(in_line)
            a,b,c,d,e,f,g,h,i = line2[0],line2[1],line2[2],line2[3],line2[4],line2[5],line2[6],line2[7],line2[8]
            #print(a)
            recover_atribut = [x for x in i.strip('\r').strip("").split(';') if x!=""]   #list to generates the attributes
            #print(recover_atribut)

            Chr = str(a)

            start_ = str(d)
            end_ = str(e)                       #get the start and end
            tag= Chr+":"+str(start_)+"-"+str(end_)      # generate name

            #if start_=='0': start_='1'              # to avoid error: 'start too small' from HTSeq--> esto esta anulado porque en realidad no se para que es

            source_= str(b)
            type_= str(c)                           #for the column 3 (type) for exon, to call expression
            score_= str(f)
            strand_= str(g)
            phase_= str(h)
```

```
        recover_atribut.insert (0,'source=%s' %source_ )
        recover_atribut.insert (1, 'type=%s' %type_ )
        #recover_atribut.insert (2, 'score=%s' %score_ )        # modify the attributes adding name
        #recover_atribut.insert (3, 'strand=%s' %strand_ )
        recover_atribut.insert (4, 'phase=%s' %phase_ )
        attributes_ = ";".join(recover_atribut)          # reconstitute the attributes_ the new attributes name

        print(Chr+"\t"+start_+"\t"+end_+"\t"+tag+"\t"+attributes_+"\n")

        #write .bed4
        #out_file_bed.write(Chr+"\t"+start_+"\t"+end_+"\t"+tag+"\t"+attributes_+"\n")

out_file_bed.write(Chr+"\t"+start_+"\t"+end_+"\t"+tag+"\t"+score_+"\t"+strand_+"\t"+attributes_+"\n)
out_file_bed.close()
print((time.time()-start_time))
```

>Extraer el fasta de los gff3

```
import time
start_time = time.time()
import subprocess
import os
from Bio import SeqIO

#### Definicion de funciones #####

##########################################
##Definition of parts from read gff3###
##########################################
```

```
def id_gff3 (inputFile):
	ltID = {}
	with open (inputFile, 'r') as infile:
		for in_line in infile:
			if in_line.startswith('#'):continue
			else:
				line2 =[x for x in in_line.strip('\n').split('\t') if x !=""]
				a,b,c,d,e,f,g,h,i =
line2[0],line2[1],line2[2],line2[3],line2[4],line2[5],line2[6],line2[7],line2[8]
				recover_atribut = [x for x in i.strip('\r').strip("").split(';') if
x!=""]   #list to generates the attribte
				atr = {}
				atr.update(zip('abcdefghijklmnopqrst', recover_atribut))
				chr_ = str(a)
				start_ = str(d)
				end_ = str(e)
				tag = chr_+":"+start_+"-"+end_

				Parent = str(atr['a']).replace('Parent=','')
				Ltr_similarity = str(atr['d']).replace('Ltr_similarity=','')
				ltID[Parent] = Ltr_similarity
	infile.close()
	return ltID

##########################################################################
############################

def tag_gff3 (inputFile):
	ltTAG = {}
	with open (inputFile, 'r') as infile:
		for in_line in infile:
			if in_line.startswith('#'):continue
			else:
				line2 =[x for x in in_line.strip('\n').split('\t') if x !=""]
```

```
                                a,b,c,d,e,f,g,h,i                                                =
line2[0],line2[1],line2[2],line2[3],line2[4],line2[5],line2[6],line2[7],line2[8]
                                recover_atribut = [x for x in i.strip('\r').strip("").split(';') if
x!=""]   #list to generates the attribte
                                atr = {}
                                atr.update(zip('abcdefghijklmnopqrst', recover_atribut))
                                chr_ = str(a)
                                start_ = str(d)
                                end_ = str(e)
                                tag = chr_+":"+start_+"-"+end_

                                Parent = str(atr['a']).replace('Parent=','')
                                Ltr_similarity = str(atr['d']).replace('Ltr_similarity=','')
                                ltTAG[tag] = Ltr_similarity
        infile.close()
        return ltTAG

#########################################################################
########################################

def TE_A (inputFile):

   te_e =[]
   with open (inputFile, 'r') as infile:
      for in_line in infile:
         if in_line.startswith('\t'): continue
         else:
            line_a = [x for x in in_line.strip('\n').split('\t') if x!='']
            a = line_a[0]
            te_e.append (a)
      return te_e

#########################################################################
##########
```

```
### path ###
root = str (os.getcwd()) #traigo el directorio en el que se trabajo


#### archivos de entradas ######

#inf_f = raw_input('Entrada Speen: ')
inf_f = 'TEs_Allnoncanonical_plus_unique_FINAL_LTRharvest_Together'
inputFile_f = '%s/%s.fasta' %(root, inf_f)

inf_l = 'Copia de Analysis_DE_Lycopersicum_tab'
inputFile_l = '%s/TE_activos/%s.txt' %(root, inf_l)

inf_p = 'Copia de Analysis_DE_Pennellii_tab'
inputFile_p = '%s/TE_activos/%s.txt' %(root, inf_p)

lis_te_a_l = TE_A (inputFile_l)
lis_te_a_p = TE_A (inputFile_p)
lis_te_a = lis_te_a_l + lis_te_a_p


lsdir = [f for f in os.listdir('clusters') if str(f).endswith('.gff3')]

#print(lsdir)

ltIDs = {}
ltTAGs = {}
fallas =[]

numf = 0
nump = 0

for x in lsdir:
```

```
    #print(type(x))
    inputFile_x = '%s/clusters/%s' %(root, x)
    ltIDs = id_gff3 (inputFile_x)
    listID =  [clave for clave, valor in ltIDs.items()]
    ltTAGs = tag_gff3 (inputFile_x)
    listTAG =  [clave for clave, valor in ltTAGs.items()]
    with open (inputFile_f, 'rU') as inputfasta:
       for record in SeqIO.parse(inputfasta, format='fasta'):
          if record.id in listID:
             out_f = x.replace('.gff3','.fasta')
             out_fasta = open ('%s/clusters/%s' %(root, out_f), 'a')
             if record.id in lis_te_a:
                out_fasta.write ('>%s/%s*\n%s\n' %(record.id,
ltIDs[record.id],record.seq))

             else:
                out_fasta.write ('>%s/%s\n%s\n' %(record.id, ltIDs[record.id],record.seq))

             out_fasta.close
             nump = nump + 1
          elif record.id in listTAG:
             out_f = x.replace('.gff3','.fasta')
             out_fasta = open ('%s/clusters/%s' %(root, out_f), 'a')
             if record.id in lis_te_a:
                out_fasta.write ('>%s/%s*\n%s\n' %(record.id,
ltTAGs[record.id],record.seq))

             else:
                out_fasta.write ('>%s/%s\n%s\n' %(record.id,
ltTAGs[record.id],record.seq))

             out_fasta.close
             nump = nump + 1
          else:
```

```
            #continue
            numf = numf + 1



print ('aciertos'+str(nump))
print ('fallas'+str(numf))

#print (cluster['0_SL3.0ch10:10159041-10167904.gff3'])
#for cla, val in cluster.items():
        #print(cla+'\n'+'-'+val)
        #out_f = cla.replace('.gff3','.fasta')
        #out_fasta = open ('%s/clusters/%s' %(root, out_f), 'a')
        #out_fasta.write (val)
        #out_fasta.close
print(time.time()-start_time)
```

>Reverse to foward

```
import time
start_time = time.time()
import subprocess
import os
from Bio import SeqIO
import csv
from Bio.Seq import Seq #aparente mente no es necesario ya que con el llamado anterior ya importa la función

#### Definicion de funciones #####

##############################################################
def list_strand (archv, root):
    plus_s = []
    reverse_s = []
```

```
    with open ('%s/%s' %(root, archv), 'r') as exel_file:
        read_file = csv.reader (exel_file, delimiter='\t')
        for in_line in read_file: # cuando lee el archivo lo lee linea por linea creando una lista de la linea donde cada elemento da la linea es una celda
            #print(in_line)
            if in_line[0]='' or in_line[0].startswith('T'): continue # salteo o ignoro el ecabazado del exel
                #print(in_line[0])
            else:
                a,b,c,d =in_line[0], in_line[1],in_line[2],in_line[3]
                tag = (str(a)).replace(' ','_')  # ajusto el tag para que sea parecido al tag del fasta
                strand = str (d)
                if str('+') == strand: #creo una lista con cada una de los sentidos de transcripcion, no quiere decir que el sentido de trancripcion que le puse sea el que realmente tiene
                    plus_s.append (tag)
                    #print(plus_s)
                else:
                    reverse_s.append (tag)
                    #print(reverse_s)
    exel_file.close()
    return plus_s, reverse_s

############################################################

def cluster_tag (archv, root):
    clust = {}
    with open ('%s/%s' %(root, archv), 'r') as exel_file:
        read_file = csv.reader (exel_file, delimiter='\t')
        for in_line in read_file:
            if in_line[0]='' or in_line[0].startswith('T'): continue
            else:
                a,b,c,d =in_line[0], in_line[1],in_line[2],in_line[3]
```

```
            tag = (str(a)).replace(' ','_')
            cluster = str (c)
            clust[tag] = cluster
    exel_file.close()
    return clust

############################################################

### Path and File ###

root = str (os.getcwd()) #directorio en el que se trabajo

arch_fasta = 'TEs_Allnoncanonical_plus_unique_FINAL_LTRharvest_Together.fasta'

arch_class_transp = 'transposones de los arboles.csv' # guardo el exel con un formato
csv que el python puede leer

output_fasta = 'fasta_search/TE_corregidos'
if os.path.exists ( output_fasta ) == False:
    os.makedirs( output_fasta, 0o777)

###################
#### Main ########
###################

plus, reverse = list_strand (arch_class_transp, root)
clust_tag = cluster_tag (arch_class_transp, root)

with open ('%s/%s' %(root, arch_fasta), 'rU') as fasta_file:
    for record in SeqIO.parse(fasta_file, format='fasta'):
        #output_fasta_c = open ('%s/%s' %(root, output_fasta), 'a')
        if record.id in plus:
            output_fasta_c = open('%s/%s/%s.fasta' %(root, output_fasta,
clust_tag[record.id]), 'a')
```

```
            output_fasta_c.write('>%s\n%s\n'%(record.id, record.seq))
            output_fasta_c.close
            #print(clust_tag[record.id])
        elif record.id in reverse:
            output_fasta_c = open('%s/%s/%s.fasta' %(root, output_fasta,
str(clust_tag[record.id])), 'a')
            #sequence = record.seq
            output_fasta_c.write('>%s\n%s\n' %(record.id,
(record.seq).reverse_complement())) # escribo el reverse comlemetary
            output_fasta_c.close
            #print(clust_tag[record.id])
#output_fasta_c.close
fasta_file.close

print(time.time()-start_time)
```