FACULTAD DE CIENCIAS AGRARIAS

FACULTAD DE CIENCIAS BIOQUÍMICAS Y FARMACÉUTICAS

“Diseño de una arquitectura en pipeline para la descarga y análisis de secuencias de promotores en *Solanum lycopersicum*”

Ing. en Sist. Alejandro Damián Pistilli Neri

TRABAJO FINAL PARA OPTAR POR EL TITULO DE ESPECIALISTA EN BIOINFORMÁTICA

Tutora: Dra. Débora Arce

2016

Diseño de una arquitectura en pipeline para la descarga y análisis de secuencias de promotores en *Solanum lycopersicum*

Alejandro Damián Pistilli Neri

Ingeniero en sistemas de información, Universidad Tecnológica Nacional

Este Trabajo Final es presentado como parte de los requisitos para optar al grado académico de Especialista en Bioinformática de la Universidad Nacional de Rosario y no ha sido presentado previamente para la obtención de otro título en esta u otra Universidad. El mismo contiene los resultados obtenidos en investigaciones llevadas a cabo en la Facultad de Ciencias Agrarias de Rosario durante el período comprendido entre el 9 de abril de 2015 y el 12 de diciembre de 2016, bajo la tutoría de la Dra. Débora Arce.

Alejandro Damián Pistilli Neri

Débora Arce

Defendida: 21 de abril de 2017.

Agradecimientos

A la Dra. Débora Arce por su paciencia y pedagogía ante mis preguntas repetitivas.

Al Dr. Guillermo Pratta por su disposición y buenos consejos.

A la Dra. Telma Scarpeci y a la Ing. Laura Angelone

A Luciana, Silvina, Araceli y Evelina quienes me acompañan a diario y me apoyan en este tiempo de crecimiento profesional.

A la cátedra de genética y al grupo de tomate .

A la Dr. Liliana Picardi y al Dr. Juan Pablo Ortiz por proponerme hacer la especialización y motivarme.

A Silvana, Rubén, Pablo y Aye por su apoyo y aguante.

A Celi, compañera del camino.

Al Dios de Jesucristo quien le da sentido a mi vida.

Resumen

Se presenta el desarrollo de una arquitectura en pipeline que automatiza la descarga de promotores de *Solanum lycopersicum* desde la Sol Genomics Network y luego los analiza con el programa MEME y TOMTOM. El código está disponible en www.github.com/lalebot/pip-prom-tom y utiliza Git como software de versionado de software. Se combina el uso de threads en Python, expresiones regulares y base de datos SQLite que conjuntamente disminuyen el tiempo de descarga de los promotores y optimiza la utilización de recursos informáticos.

La presente metodología es potencialmente aplicable a otras áreas biológicas.

Abstract

Design an instruction pipeline for download and analysis of promoter sequences in *Solanum lycopersicum*

This work presents the development of an instruction pipeline that automates the download of *Solanum lycopersicum* promoters from the Sol Genomics Network and then analyzes them using the MEME and TOMTOM programs. The code is available at www.github.com/lalebot/pip-prom-tom and uses Git as software versioning. It combines the use of threads in Python, regular expressions and SQLite database, all of which reduce the download time of the promoters and optimize the use of computer resources. This methodology is potentially applicable to other biological areas.

Índice

Introducción

Las nuevas tecnologías de secuenciación han permitido la generación de grandes cantidades de información que está siendo recopilada en bases de datos y en diferentes plataformas bioinformáticas. Esta información está disponible para la comunidad científica y consta de secuencias génicas, estructura y localización de genes, herramientas para visualización y manipulación de secuencias de ADN, ARN y proteínas. Además contienen resultados de experimentos de transcriptómica, tales como secuenciación del ARNm (RNA-Seq) y microarreglos.

Las bases de datos y plataformas más relevantes para las Solanáceas y específicamente para el tomate, son actualmente: Sol Genomics Network (SGN, https://solgenomics.net), Tomato Functional Genomics Database (http://ted.bti.cornell.edu/) (http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3013811/), Tomato Genomic Resources Database (http://59.163.192.91/tomato2/) (Suresh et al, 2014). La plataforma SGN es un repositorio primario de datos fenotípicos, genéticos, genómicos, de expresión génica y metabólica proveniente de la familia de las Solanáceas y otras especies relacionadas. SGN almacena los datos de secuenciación del genoma de la variedad Heinz 1706 de *S. lycopersicum* como así también los de otras variedades y cultivares silvestres (ej. *S. pennelli* LA0716, *S. pimpinellifolium* LA1789). En esta plataforma reside una gran variedad de herramientas bioinformáticas que permiten la manipulación de estos *datasets* (Tomato & Consortium, 2012), posibilitando así que los genomas secuenciados sirvan de referencia para el estudio de otras variedades para las cuales aún no hay datos de secuenciación de nueva generación. Esta información ha comenzado a ser utilizada para el mejoramiento de la calidad de los frutos de tomate en variedades locales de nuestro país (Cambiasso et al, 2015), a veces presentándose diversas limitaciones para el usuario en el manejo de las herramientas disponibles en este tipo de plataformas.

La conversión desde el estado de madurez del fruto de tomate (EMT) verde al estado completamente maduro rojo implica cambios dramáticos en el color, composición, aroma, sabor y textura del fruto. La maduración es un proceso que incluye alteraciones en el metabolismo y la expresión de genes, teniendo un efecto dramático en la calidad de los frutos. Los diversos EMT como así también la respuesta al estrés o diferentes situaciones fisiológicas

vegetales, implican a nivel molecular, una reprogramación y una modificación de la regulación de la expresión de genes o grupos de genes específicos (Gierson y Kader, 1986). Los mecanismos implicados en la regulación de la expresión génica abarcan la interacción de una proteína o factor de transcripción (del inglés, *transcription factor*, TF) con una secuencia corta (5-15 pares de bases o pb) o motivo de ADN. Los motivos de ADN se encuentran en los promotores génicos y su identificación ha constituido por años un tópico de discusión en el ámbito científico. Un promotor es la región del genoma cercana al sitio de inicio de la transcripción (SIT) de un gen y generalmente se la ubica 1000 pb río arriba (*upstream*). Algunos autores describen aproximadamente 200 río abajo (*downstream*) del SIT, tales como fue reportado por Klug y Cummings, M.R. (2003). La interacción entre los TFs y sus motivos en el ADN es específica y lleva a la inducción o represión de la expresión génica (Lin, 2012.; Zambelli, Pesole, & Pavesi, 2013). Numerosos métodos han sido utilizados para identificar motivos en secuencias promotoras. Su identificación ha sido uno de los problemas más ampliamente estudiados, no sólo por su significado biológico sino también por su dificultad bioinformática. Esta dificultad deriva de la cantidad enorme de datos con los que se cuentan en la actualidad y de la necesidad de acceso a los mismos de formas flexibles y parametrizables para su análisis. Frecuentemente, es necesario analizar grupos de promotores provenientes de genes co-regulados, es decir aquellos genes que poseen un perfil de expresión similar, ya sea por inducción o represión en la expresión génica. La disponibilidad de herramientas bioinformáticas para el análisis de promotores dentro de las Solanáceas (tomate, papa, tabaco, entre otras) es baja, con ausencia de interfaces gráficas adecuadas para analizar grupos de genes simultáneamente. Tal es así que este trabajo se originó a partir de necesidades o preguntas biológicas por resolver, tal como se observa en dos Trabajos Finales de Especializacón en Bioinformática (Gismondi, 2016; Arce, 2016). En estos trabajos se contempló la necesidad de analizar promotores de genes de interés para la calidad de frutos de tomate y durazno, buscando motivos sobre-representados o bien construyendo bases de datos con motivos de interés.

En SGN se encuentra disponible una herramienta para la descarga de promotores con ciertas limitaciones, como la imposibilidad de obtener promotores de varios genes en forma simultánea. Si bien existe la posibilidad de descargar en *bulk* las zonas promotoras de todos los genes de *S. lycopersicum* cv Heinz 1706, estos zonas sólo poseen una cantidad de pb

upstream/downstream del SIT prefijadas por SGN (https://solgenomics.net/tools/bulk?mode=ftp) y no permiten la flexibilidad en la selección de los parámetros asociados.

Por tal motivo, el presente trabajo tiene como objetivo principal desarrollar una herramienta que se conecte con la web de SGN, para la identificación, extracción y posterior análisis de secuencias de grupos de promotores con los programas MEME (Bailey y Elkan, 1994; http://www.sdsc.edu/~tbailey/papers/ismb94.pdf) y TOMTOM (Gupta, Stamatoyannopoulos, Bailey & Noble, 2007). MEME es un algoritmo que identifica uno ó más motivos en una colección de secuencias de ADN o proteínas. El archivo de salida de MEME es matricial y puede ser comparado contra bases de datos matriciales correspondientes a motivos conocidos Ej. Jaspar DNA CORE (2016) (Mathelier et al, 2015 y Franco-Zorrilla et al, 2014). Posteriormente, mediante el uso del algoritmo TOMTOM, se cuantifica estadísticamente la similitud entre el motivo incógnita o *query* obtenido por MEME y el motivo anotado en la base de datos de motivos conocidos. Así es como MEME sirve para identificar motivos sobrerrepresentados en una lista de secuencias promotoras y TOMTOM permite clasificar estos motivos de ADN según sea su unión con diferentes familias de TFs específicos.

Por todo esto, el diseño de una interfaz que permita descargar promotores a partir de una lista, estableciendo un tamaño de secuencia promotora *upstream/downstream* al SIT parametrizable, y utilizando las coordenadas de *S. lycopersicum* cv Heinz 1706, posibilitará el análisis de múltiples secuencias promotoras en simultáneo.

Objetivo

Desarrollar una *arquitectura en pipeline* para la descarga y análisis de promotores de genes de *Solanum lycopersicum* cv Heinz 1706, desde la web Sol Genomics Network (http://solgenomics.net/).

Los promotores descargados serán organizados en un único archivo FASTA para su posterior análisis con los programas MEME y TOMTOM.

Materiales y Métodos

La *arquitectura en pipeline* permite ir transformando un flujo de datos en un proceso comprendido por varias fases secuenciales, siendo la entrada de cada una la salida de la anterior.

Se desarrolló un *script* con las órdenes de procesamiento en el lenguaje de programación Python versión 3 (http://www.python.org) que funciona sólo bajo sistemas operativos GNU/Linux y derivados. Un *script* contiene órdenes que el programa intérprete lee una a una y las ejecuta secuencialmente. Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Es un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Cuenta con una gran comunidad de usuarios y desarrolladores. Existe un proyecto llamado Biopython (Cock et al, 2009) que consiste en un conjunto de bibliotecas para Python, especializadas para cuantificar y hacer cálculos con datos biológicos, programados por una comunidad internacional (https://github.com/biopython/biopython.github.io/).

El seguimiento y versionado del proyecto se realizó con Git (http://git-scm.com/). Git es un software de control de versiones diseñado por Linus Torvalds, uno de los creadores del sistema operativo GNU/Linux. El proyecto está alojado en la web GitHub (https://github.com/lalebot/pip-prom-tom) y es de libre acceso. En la misma se permite su clonación, duplicación y posterior modificación por cualquier usuario de la web. También contiene el manual de instalación y uso en el archivo README.md (https://github.com/lalebot/pip-prom-tom/blob/master/README.md) que incluye:

- Requisitos para la ejecución del *script*.
- Instructivo para la descarga del *script*.
- Detalle de los parámetros del archivo de configuración.
- Detalle de los comandos que el *script* necesita para ejecutarse.

Resultados y Discusión

El presente Trabajo Final adjunta un DVD que incluye el código completo del programa, manual de instalación y uso, y un video en donde se muestra al programa corriendo. Esto permite al usuario y a los lectores (público en general y evaluadores) de este Trabajo Final replicar los resultados obtenidos y correr el *script* sin necesidad de conectarse.

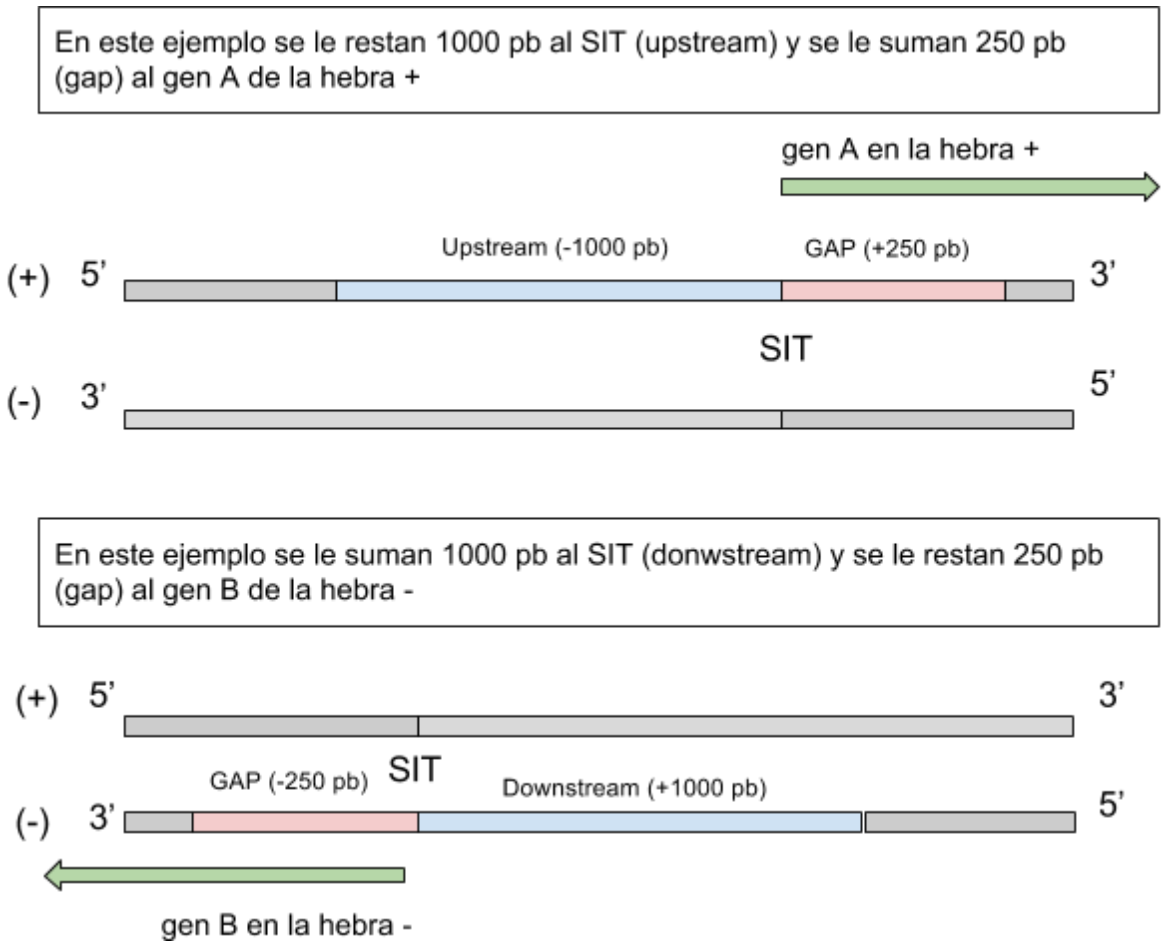
Resultados

Entradas del pipeline:

- Una lista con los códigos de promotores de *S. lycopersicum* para descargar de SGN.
- Un archivo de configuración que contiene los parámetros personalizables para la ejecución del *script* y para la corrida de los algoritmos MEME y TOMTOM.

Al ejecutar el *script*, se debe señalar el nombre del archivo que contiene la lista de promotores, el nombre que se le va a dar a ese proyecto, si van a descargarse los promotores *upstream* o *downstream*, el gap y si el *script* se ejecuta en modo pipeline o no.

En el presente trabajo nos referimos como gap a la cantidad de pares de bases extras que se descargarán contabilizando las bases a partir del SIT de cada gen. Si el gen está en la hebra (+), se suman las bases extra al SIT, y si el gen se encuentra en la hebra (-), estas bases extras se restan (Esquema 1).



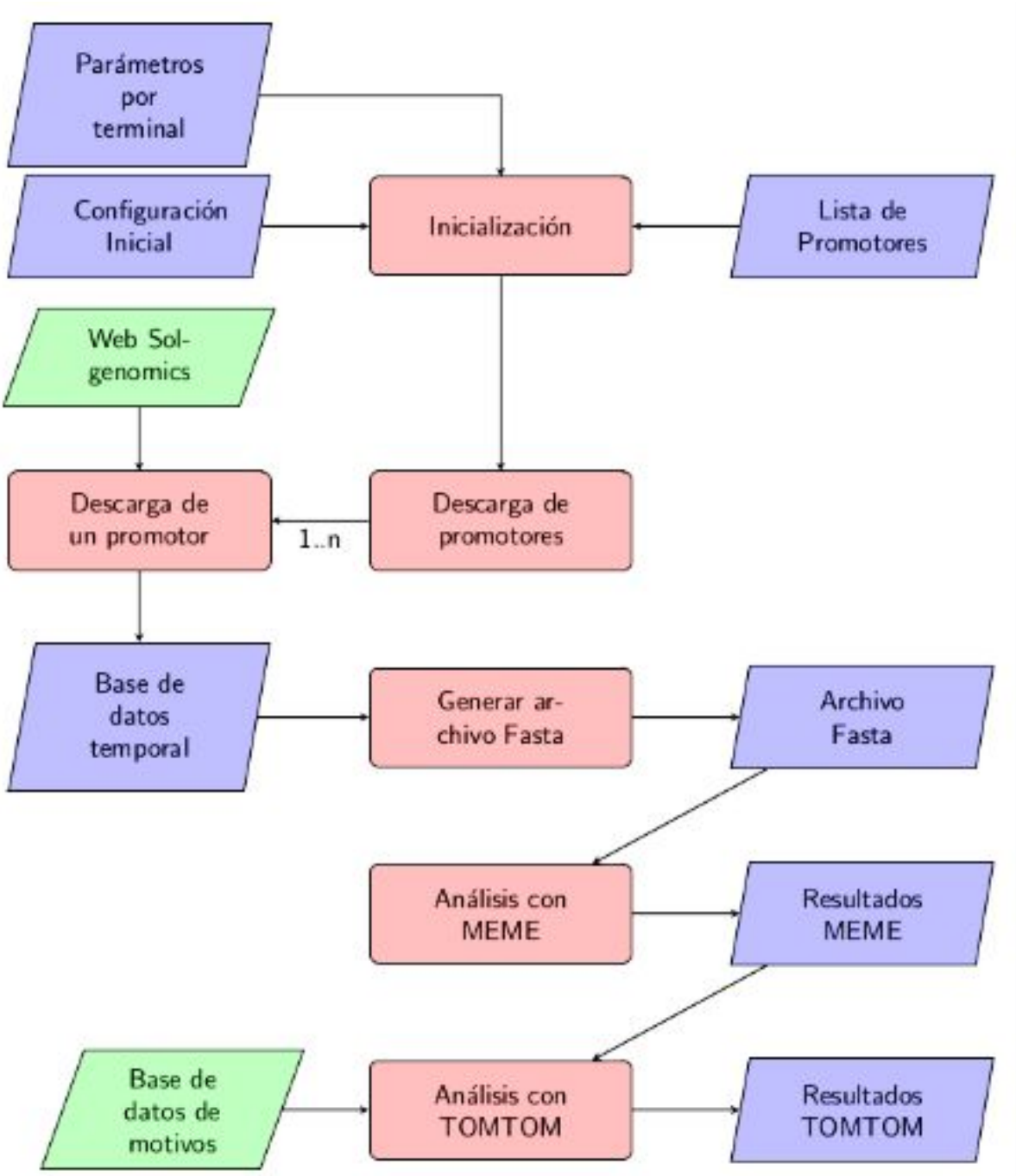
Esquema 1: Obtención de la secuencia promotora, contando a partir del SIT para los genes A (con región codificante CDS en hebra positiva) y B (con CDS en hebra negativa)

El modo pipeline del *script* no necesita interacción del usuario sino que ejecuta todos los pasos de manera secuencial.

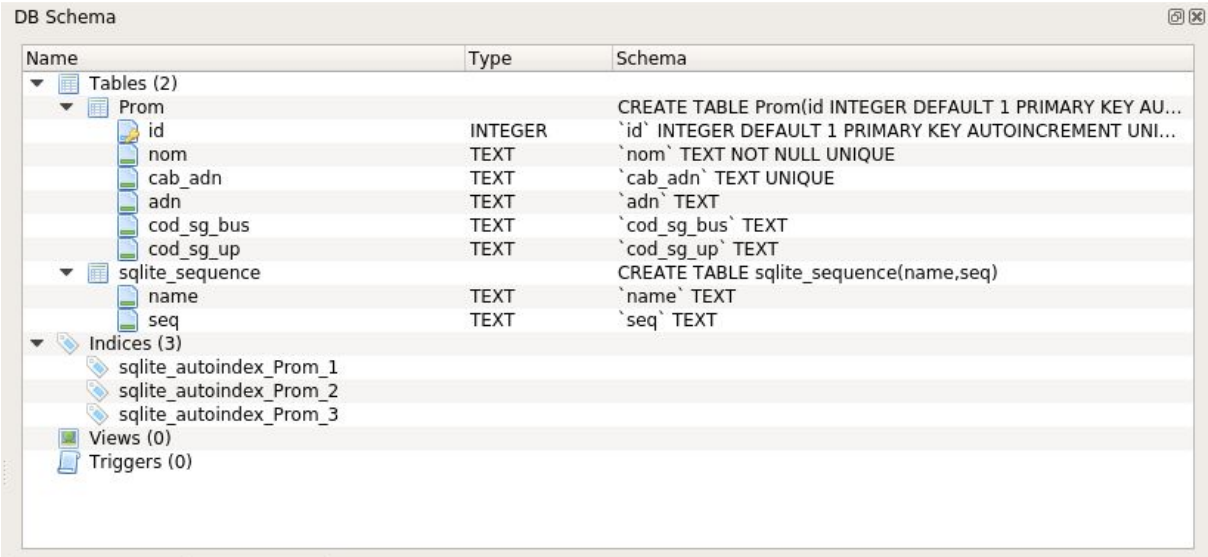
Procesamiento del pipeline (Esquema 2):

- A partir de una lista de códigos de promotores de SGN, el *script* comienza a hacer consultas para obtener los promotores a través de threads (hilos), que trabajan en paralelo para la optimización del rendimiento y ancho de banda de internet. La cantidad de threads se define en el archivo de configuración inicial (*conf.ini*). Este archivo puede modificarse antes de cada ejecución del *script*.

- Cada uno de esos hilos o procesos hijos, descarga el código html del resultado de una búsqueda en SGN. Dentro de ese html, el *script* realiza la búsqueda de un código único que identifica al promotor dentro de SGN para realizar otra consulta y descarga el código html de la misma. Las búsquedas dentro de los códigos html se realizan con expresiones regulares.
- Una vez que el *script* tiene el código de identificación del promotor, realiza otra consulta para obtener el html en donde encuentra el SIT que sirve de referencia para hacer la descarga del promotor.
- Con el SIT encontrado se realiza la descarga de los pares de bases de ese promotor en base a los parámetros *downstream/upstream* y *gap*.
- El resultado obtenido del promotor se guarda en una base de datos SQLite que contiene todos los datos descargados y procesados (Esquema 3). En el caso de que el proceso de descarga tenga una interrupción, el *script* puede retomar las búsquedas a partir de los datos incompletos de esta base de datos. La base de datos permite que todos los hilos estén leyendo y escribiendo a la vez sin que se pierda ningún dato ya que la misma gestiona el acceso simultáneo de varios procesos.
- Cada vez que el hilo encuentra y descarga correctamente un promotor realiza una nueva consulta a la base de datos para continuar la búsqueda de otro promotor.
- Una vez que los hilos han cargado toda la base de datos completando las cadenas de promotores faltantes, el *script* genera un archivo FASTA que contiene todas las cadenas de promotores.
- Este archivo sirve de entrada para que ser analizado por el programa MEME. Los parámetros de análisis que utiliza MEME son tomados del archivo *conf.ini.*
- El *script* descarga la última versión de base de datos de motivos que necesita el programa TOMTOM.
- Los resultados del algoritmo MEME sirven de entrada al programa TOMTOM que los compara con la base de datos de motivos de ADN conocida. La base con la cual se hace la comparación está establecida en el archivo *conf.ini*. Por defecto es la base "Jaspar DNA CORE (2014)" pero puede ser modificada con otras bases de datos (por ej Franco- Zorrilla, 2014).
- Si un error sucede durante el procedimiento, el mismo queda registrado en un archivo de *logs.log*



Esquema 2: Arquitectura en pipeline



Esquema 3: Esquema de la base de datos temporal

Descarga y ejecución del *script*

Los manuales de descarga, instalación y uso se encuentran en: https://github.com/lalebot/pip-prom-tom

Los requisitos para la ejecución del *script* son:

- Tener una distribución basada en GNU/Linux y los siguientes paquetes instalados:
    - Una terminal de Linux.
    - **Python versión 3**: para poder correr el *script*.
    - **Git**: para poder clonar el repositorio en donde está el *script* y enviar cambios al mismo.
    - **SQLite versión 3**: para gestionar la base de datos donde se guardan las cadenas de los promotores.
    - **wget**: para realizar descargas desde la web.
    - **tar**: para desempaquetar archivos comprimidos.
    - **MEME-SUITE**: Para correr los programas MEME y TOMTOM

Instalación de requisitos en Debian, Ubuntu y derivados:

```
$ apt-get install git python3 sqlite3 wget tar
```

Instalación de requisitos en Archlinux y derivados:

```
$ pacman -Sy git python3 sqlite3 wget tar
```

Una vez instalados, hay que descargar el proyecto del *script* con el comando "**git clone**" desde una terminal. El mismo permite hacer una clonación del repositorio del proyecto que se encuentra en la web GitHub.

```
$ git clone http://github.com/lalebot/pip-prom-tom.git
```

La clonación crea una carpeta llamada *"pip_prom_tom"* que contiene todo el proyecto. Abrimos ese directorio:

```
$ cd pip-prom-tom
```

Cada uno de los parámetros que el *script* requiere se conforma de un guión medio y una letra que identifica al parámetro.

Un ejemplo de ejecución del *script*:

```
$ python3 pip_prom_tom.py -i list_prom.txt -o proyecto1 -u 1000 -g 250 -p 1
```

Donde:

- **python3** llama al intérprete de Python para que ejecute el archivo que se escribe a continuación.
- **pip_prom_tom.py** es el *script* escrito en Python.
- A través del parámetro **-i** (input) se indica al *script* el nombre del archivo de entrada que contiene la lista de promotores a descargar y analizar. El archivo debe encontrarse en el mismo directorio que el *script*. Generalmente es un archivo de texto plano.
- Con el parámetro **-o** (output) se indica el nombre del proyecto de salida. Este nombre se usa para crear una carpeta que contiene todos los archivos que genera el *script* y los archivos temporales que usa.
- El parámetro **-u** (upstream) establece la cantidad de pares de bases upstream que se van a descargar de cada promotor. En el caso de descargar los pares de base downstream se usa el parámetro **-d** (downstream) en vez de **-u**.
- El parámetro **-g** establece la cantidad de pares de bases extras que se van a descargar en el sentido contrario del SIT. Por defecto es cero.
- El *script* puede ejecutarse en modo pipeline (**-p 1**) o no (**-p 0**). El modo pipeline no requiere la interacción con el usuario. Por defecto el *script* no se ejecuta en modo pipeline.

Los resultados se almacenan en una sub-carpeta dentro del mismo directorio que tiene como nombre el nombre del proyecto que se establece en la ejecución del *script* más la extensión "**_out**".

Para obtener ayuda de los parámetros necesarios y los admitidos hay que ejecutar el *script* y agregar **-h** (help):

```
$ python3 pip_prom_tom.py -h
```

La salida es:

```
Usage: pip_prom_tom.py [options]
Options:
 -h, --help      show this help message and exit
```

```
-i FILE, --in=FILE   Archivo de entrada
-o DIROUT, --out=DIROUT Proyecto de salida (default="proy")
-p PIPE, --pip=PIPE   Modo pipeline (default=0)
-u UP, --up=UP     Cantidad bp upstream (default=0)
-d DOWN, --down=DOWN  Cantidad bp downstream (default=0)
-g GAP, --gap=GAP    Gap de bp upstream o downstream (default=0)
```

Contenido de los archivos del proyecto (Figura 1)

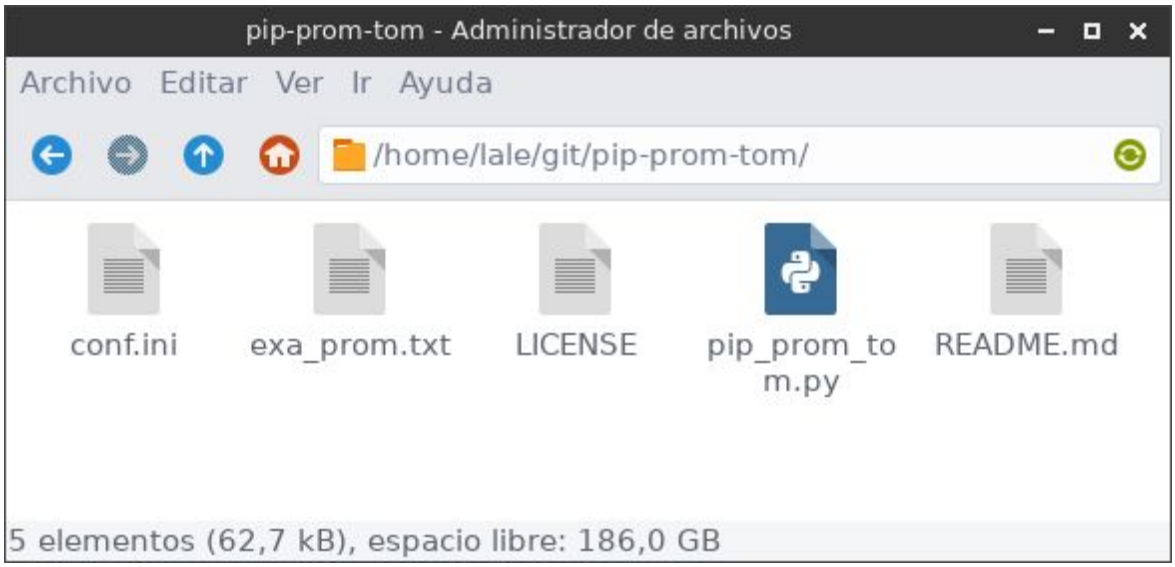


Figura 1: Archivos que contiene el proyecto

LICENSE

Contiene el tipo de licencia del proyecto. En nuestro caso es una licencia GNU GENERAL PUBLIC LICENSE versión 3 (https://www.gnu.org/licenses/gpl-3.0.html)

README.md

Contiene los requisitos para la ejecución del *script*, instrucciones para la descarga del *script*, detalle de los parámetros del archivo de configuración y comandos que el *script* necesita para ejecutarse correctamente.

conf.ini

Es el archivo de configuración inicial que contiene los valores iniciales de las siguientes seis variables. Su contenido es el siguiente:

```
threads = 20
meme-path = /usr/bin/meme-meme
meme-param = -dna -mod oops -w 8 -minw 8 -maxw 12 -maxsize 1000000000 -oc
tomtom-path = /usr/bin/meme-tomtom
tomtom-param = -min-overlap 5 -dist pearson -evalue -thresh 10 -no-ssc
tomtom-bd = motif_databases/JASPAR/JASPAR_CORE_2014_plants.meme
```

Cada una de las variables contiene un valor que el *script* requiere para su funcionamiento y puede ser modificada antes de la ejecución del *script*:

- *threads*: Es la variable que establece la cantidad de threads (hilos o procesos hijos) que se ejecutarán en paralelo para descargar cada uno la secuencia de un promotor.
- *meme-path*: Esta variable contiene la dirección donde se encuentra el programa MEME.
- *meme-param*: Contiene los parámetros con los que se va a ejecutar el MEME.
- *tomtom-path*: Contiene la dirección donde se encuentra el programa TOMTOM.
- *tomtom-param*: Contiene los parámetros con los que se va a ejecutar el TOMTOM.
- *tomtom-bd*: Contiene la dirección relativa de la base de datos que se utilizará para la comparación.

exa_prom.txt

Contiene una lista de códigos de promotores de ejemplo.

pip_prom_tom.py

Es el *script* propiamente dicho, contiene más de 500 líneas de código escrito en Python.

Ejemplo

Se toma como ejemplo la lista de códigos de promotores que se encuentra en el archivo *exa_prom.txt* que se encuentra dentro del repositorio de GitHub.

Se ejecuta el siguiente comando para comenzar la descarga y análisis (Figura 2).

```
$ python3 pip_prom_tom.py -d 1000 -i exa_prom.txt - o Ejemplo
```

En este ejemplo el nombre del proyecto es “Ejemplo”.

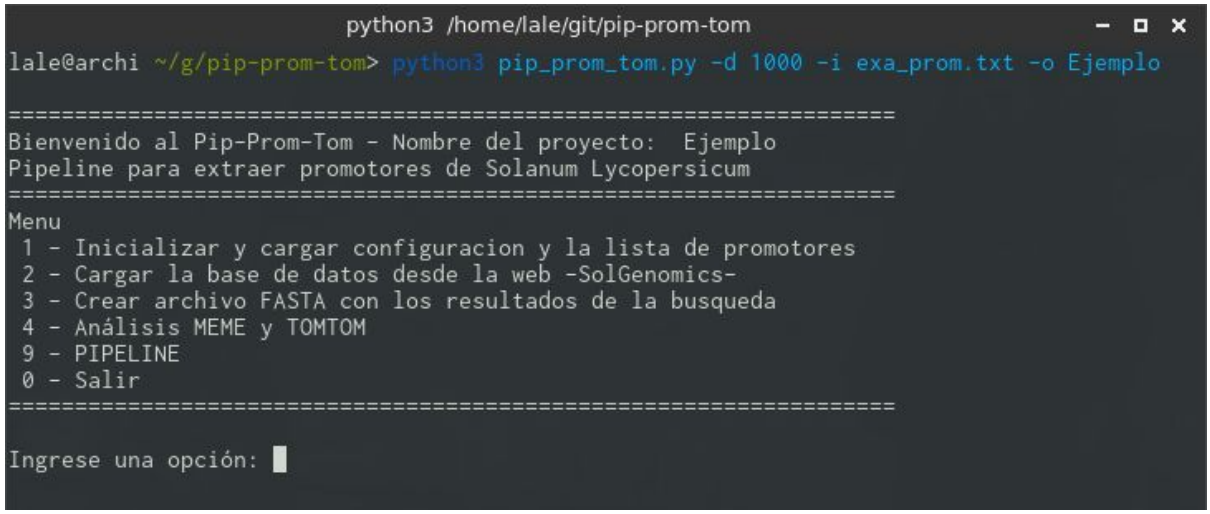


Figura 2: Menú de inicio del script

Se ingresa la opción 1 y el *script* carga los códigos de promotores a descargar (Figura 3).

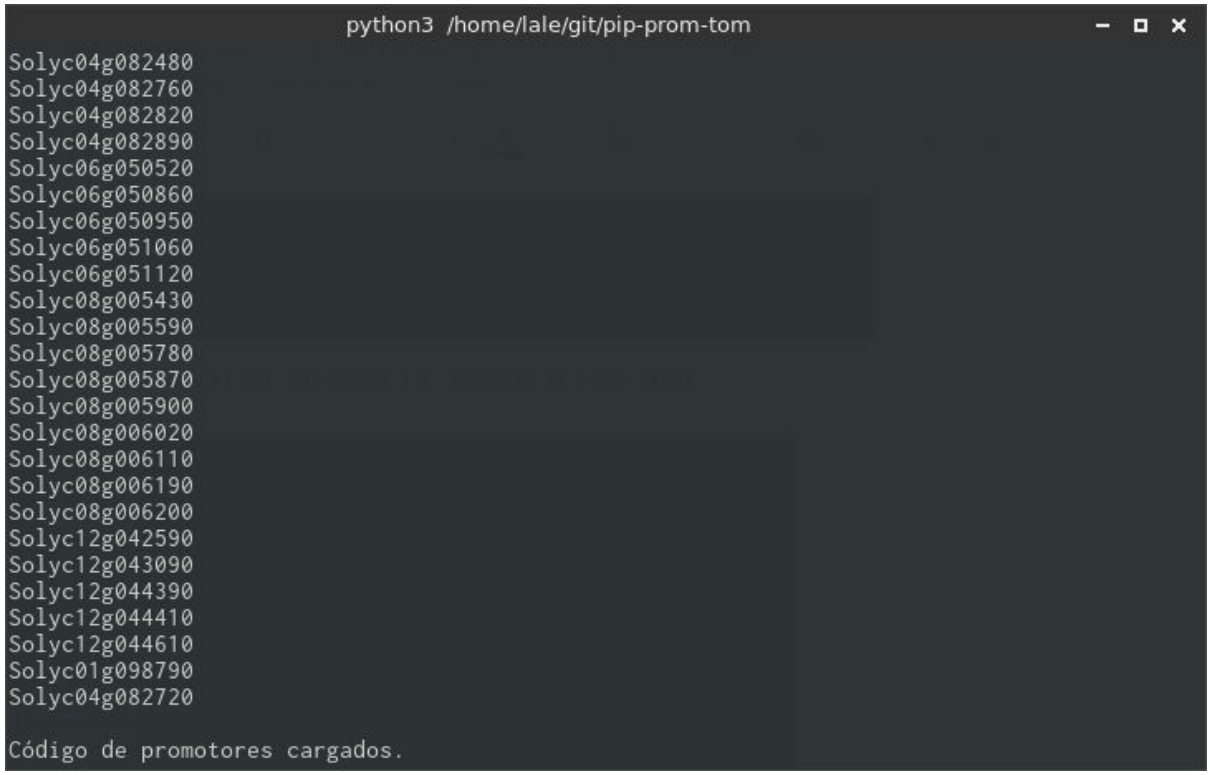


Figura 3: Ejecución de la opción 1, carga de la base de datos temporal

Luego se elige la opción 2 en donde el *script* descarga los promotores desde SGN. El *script* lanza los threads, en este ejemplo 20, en paralelo y comienza la carga de promotores en la base de datos SQLite (Figura 4).

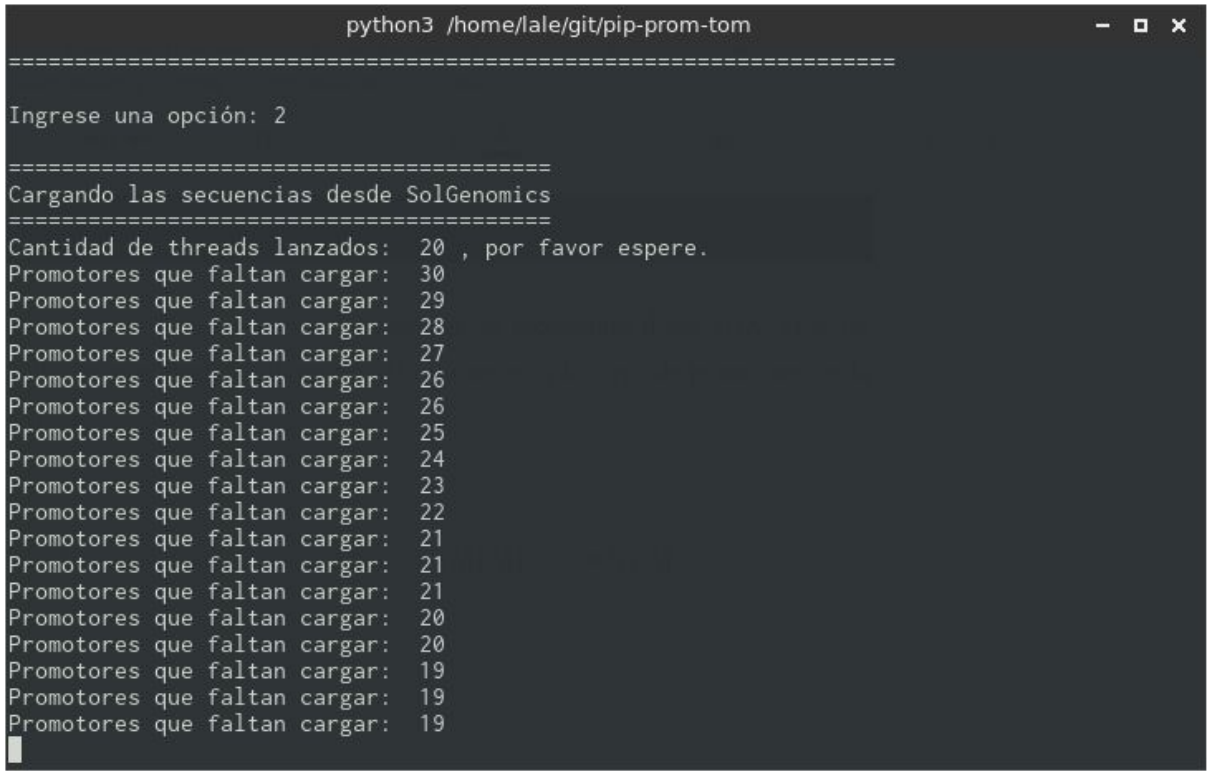


Figura 4: Ejecución de la opción 2, descarga de los motivos desde SGN

Luego se elige la opción 3 para crear el archivo FASTA con el resultado de todos los promotores (Figura 5).

Figura 5: Ejecución de la opción 3 para la creación del archivo FASTA

La opción 4 ejecuta el software de análisis MEME y continúa el análisis con el programa TOMTOM y para ello el *script* descarga la última base de datos de motivos y hace la comparación con los resultados arrojados por el MEME (Figura 6).

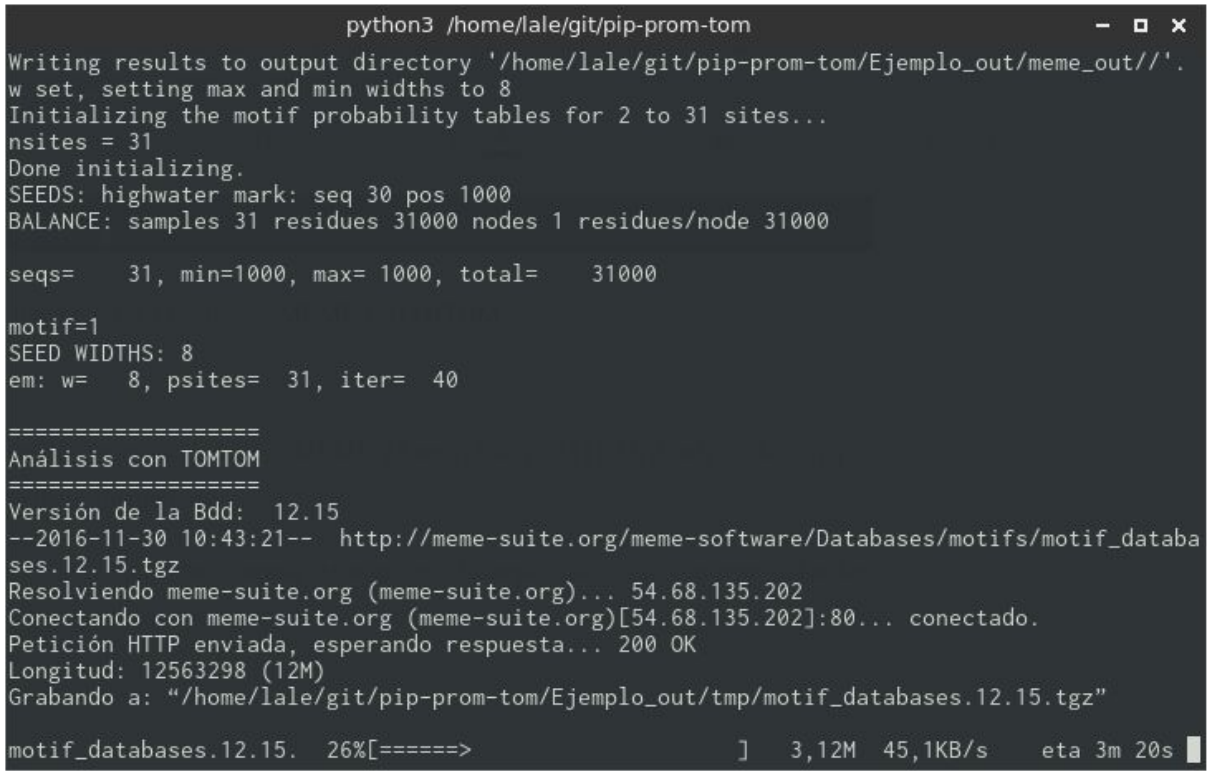


Figura 6: Ejecución del opción 4, análisis MEME y TOMTOM

Como resultado se genera una carpeta de nombre *"Ejemplo_out"* que contiene todos los archivos que el *script* generó como salida (Figura 7).

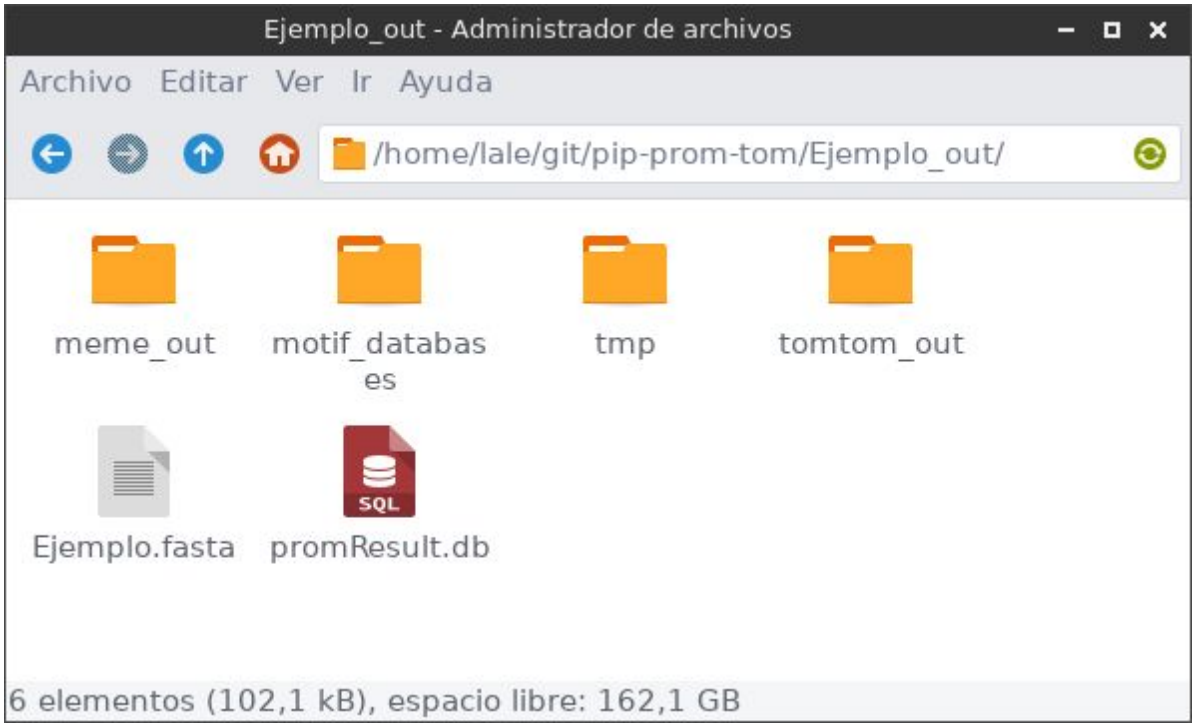


Figura 7: Contenido de la carpeta con los archivos resultantes del script

Con la opción -h el *script* muestra la ayuda sobre los parámetros que necesita el correr (Figura 8).

Figura 8: Ayuda del script

Se adjunta un video del ejemplo anterior dentro del DVD anexado al presente Trabajo Final.

Discusión

Características y funcionamiento

La arquitectura en pipeline está desarrollada con herramientas código abierto:

- GNU/Linux OS: Sistema operativo.
- Python: Lenguaje de *scriping*.
- Git: Sistema de versionado.
- SQLite: Base de datos.

Los programas de código abierto son distribuidos y desarrollados libremente; sobre los que se tiene acceso al código fuente. “Código abierto” no significa necesariamente que el software sea gratuito, sino que ofrece la libertad de poder modificar la fuente del programa, ya que muchas empresas de software encierran su código, ocultándolo y restringiéndose los derechos a sí misma.

Las ventajas de utilizar software de código abierto son varias:

- Acceso libre a cada una de las líneas de código fuente del programa.
- Ofrece la posibilidad de conocer potencialidades y limitaciones del programa en profundidad.
- En su gran mayoría, los programas de código abierto son libres y gratuitos.
- La disponibilidad del código fuente hace posible que usuarios, programadores y empresas se involucren en el desarrollo de las aplicaciones. De esta forma, el proceso de detección y corrección de errores se lleva a cabo de forma eficiente, así como la implementación de nuevas características.
- Es posible llevar a cabo modificaciones a los programas con el fin de adaptarlos a las necesidades específicas.
- Al utilizar programas de código abierto, existe independencia con respecto de una empresa específica para las tareas de mantenimiento, posibilitando la libre contratación de cualquier individuo/equipo de trabajo que posea la habilidad y el conocimiento necesario para llevar a cabo tal fin.

Impacto y ventajas del proyecto

Mediante el presente proyecto se brinda mayor flexibilidad en el uso de herramientas para la extracción y el análisis de secuencias de promotores en *S. lycopersicum* cv. Heinz 1706. El uso de este pipeline puede extenderse a otras especies de Solanáceas o variedades de tomate secuenciados. Para el caso de aquellos genomas que se hallan secuenciados de manera incompleta o parcialmente (genomas borrador) bastaría con modificar ciertas características del *script* para adaptarlo a sus nuevos usos. Por otro lado, y en relación al área biológica en la que impacta, la identificación de los motivos de ADN para la unión de TFs en familias de genes vinculadas con el proceso de maduración del fruto de tomate u otras situaciones fisiológicas, del desarrollo o bien vinculadas con la respuesta al estrés abiótico o biótico, permiten avanzar en el área de mejoramiento de la calidad y sabor de los frutos de tomate.

El presente proyecto propone una modalidad de trabajo que puede ser replicada en otras áreas de desarrollo de herramientas bioinformáticas en donde se necesite una mecanización de descargas y análisis de datos en base a una lista de búsqueda.

Ventajas de la modalidad del presente trabajo podemos encontrar que:

- Utiliza Git, un sistema de control de versionado, que. El mismo permite obtener la última versión de un proyecto para su clonación y posterior modificación.
- Explora la creación de un *script* de arquitectura en pipeline, ya que al utilizar el sistema de control de versionado Git, no sólo está disponible el proyecto en su versión final sino en todas y en cada una de las etapas del mismo desde sus inicios.
- Se almacena en GitHub.com, una de las plataformas de desarrollo colaborativo más importantes de la actualidad. En la misma se pueden encontrar proyectos colaborativos de la talla de Python, BioPython, el kernel Linux, Bootstrap, Node.js, JQuery o Ruby on Rails.
- Se aplican conceptos de escalabilidad para brindar flexibilidad y adaptación a cambios.
- Tanto el lenguaje de programación, el sistema operativo y el sistema de versionado son de código abierto y de libre uso.
- El proyecto está publicado bajo la licencia GNU General Public License (GPL) 3.0, la cual exige la publicación del código fuente y que todos los trabajos derivados del

original conserven la misma licencia GPL, no permite enlaces con módulos privativos (de código cerrado) y requiere que todos los cambios realizados a la versión original sean reflejados en el código fuente con sus respectivos autores.

- El *script* utiliza *threads* para realizar las búsquedas de los promotores. Los threads son hilos de ejecución o subprocesos de un proceso padre que permiten mejorar el rendimiento. Se utilizan en procesos de hackeo y en análisis heurísticos. Gracias a esta automatización en paralelo el resultado se obtiene cerca de diez veces más rápido que si no se usaran los mismos.
- Se usan expresiones regulares para el análisis de los códigos html y la descarga de la base de datos de motivos que utiliza el TOMTOM.

Una posible aplicación del pipeline generado en el presente Trabajo Final, sería automatizar la descarga de promotores para genes vinculados a la protección frente al daño por frío en durazno, tal como fue planteado en el trabajo del Lic. Mauro Gismondi "Estudio in silico de la expresión génica relativa a factores protectores frente al daño por frío de duraznos" (Gismondi, 2016). Asimismo, se están llevando a cabo análisis de promotores utilizando el presente pipeline, para explicar los perfiles de expresión de los genes sHSPs durante la maduración de tomate. Estos resultados están siendo abordados por los grupos de investigación del IICAR (bajo de dirección del Dr. G. Pratta) y del CIFASIS (bajo la dirección de la Dra. E. Tapia).

Conclusiones

Como resultado del presente trabajo fue posible:

Desarrollar una arquitectura en pipeline, en el lenguaje Python, que permita la automatización de la descarga de promotores de *Solanum lycopersicum* desde SGN para su posterior análisis con MEME y TOMTOM.

La creación del proyecto en una plataforma de versionado de software (www.github.com/lalebot/pip-prom-tom). Git permite guardar cada paso del desarrollo del proyecto. Dentro del mismo se encuentran los manuales de instalación y uso.

Implementación conjunta de threads en Python, expresiones regulares y base de datos SQLite para mejorar el rendimiento **disminuyendo diez veces el tiempo** necesario para una ejecución.

Adaptar e implementar una metodología que es aplicable a cualquier área biológica siempre que haya un genoma disponible anotado y disponible en la web.

Glosario

GNU/Linux (https://www.gnu.org/): El término GNU/Linux se usa para referirse a la combinación del núcleo o kernel libre denominado Linux con el sistema operativo GNU. Todo su código fuente puede ser utilizado, modificado y redistribuido libremente por cualquiera bajo los términos de la GPL (Licencia Pública General de GNU) y otra serie de licencias libres.

Arquitectura en pipeline: Consiste en ir transformando un flujo de datos en un proceso comprendido por varias fases secuenciales, siendo la entrada de cada una la salida de la anterior.

Script: Contiene órdenes que el programa intérprete las lee una a una y las ejecuta una a una de forma secuencial. Generalmente se almacena en un archivo de texto plano, lo cual facilita su edición.

Python (http://www.python.org/): Es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es administrado por la Python Software Foundation. Posee una licencia de código abierto, denominada Python Software Foundation License, que es compatible con la Licencia pública general de GNU a partir de la versión 2.1.1.

Git (https://git-scm.com/): Es un software de control de versiones diseñado por Linus Torvalds, uno de los creadores del kernel (núcleo) Linux.

GitHub (https://github.com/): Es una plataforma de desarrollo colaborativo para alojar proyectos utilizando el sistema de control de versiones Git. En la misma se pueden encontrar proyectos colaborativos de la talla de Python, BioPython, el kernel Linux, Bootstrap, Node.js, JQuery o Ruby on Rails.

Sol Genomics Network (SGN, https://solgenomics.net/): Es una base de datos "clade-oriented" dedicada a la familia de Solanáceas que incluye un gran número de familias

estrechamente relacionadas y muchas especies agronómicamente importantes como el tomate, la patata, el tabaco, la berenjena, la pimienta y la petunia ornamental híbrida.

MEME (http://meme-suite.org/tools/meme): Es un algoritmo que identifica uno ó más motivos en una colección de secuencias de ADN o proteínas.

TOMTOM (http://meme-suite.org/doc/tomtom.html): Es un algoritmo que permite clasificar motivos de ADN según sea su unión con diferentes familias de FTs específicos.

Bibliografía

Arce, D.P., Krsticevic, K.J., Ezpeleta, J., Ponce, S.D., Pratta, G.R. y Tapia, E. (2015). Heterogeneous expression pattern of tandem duplicated sHsps genes during fruit ripening in two tomato species. XX Congreso Argentino de Bioingeniería y IX Jornada de Ingeniería Clínica SABI2015. San Nicolás, Buenos Aires.

Arce, D.P. (2016). Análisis in-silico de la expresión de genes sHsps en frutos de tomate Solanum lycopersicum, Trabajo Final presentado para optar al grado académico de Especialista en Bioinformática. Disponible en Biblioteca de la Facultad de Ciencias Agrarias UNR.

Bailey, T.L., Boden, M., Buske, F. a, Frith, M., Grant, C.E., Clementi, L. and Noble, W.S. (2009). MEME SUITE: tools for motif discovery and searching. Nucleic acids research, 37(Web Server issue), W202–8. doi:10.1093/nar/gkp335

Cambiaso, V., Pereira da Costa, J.H., Rodríguez, G.R., Pratta, G.R, Picardi, L.A. Francis, D. M. y Zorzoli, R. (2015). Polimorfismo en la secuencia genómica completa entre un cultivar argentino y un especie silvestre de tomate (Solanum SPP.) Cátedra de Genética, Facultad de Ciencias Agrarias, Universidad Nacional de Rosario, Zavalla, Santa Fe, Argentina. XLIV Congreso Argentino de Genética-13 al 16 de septiembre de 2015 - Mar del Plata, Buenos Aires, Argentina.

Franco-Zorrilla, J.M., López-Vidriero, I., Carrasco, J.L., Godoy, M., Vera, P. and Solano, R. (2014). DNA-binding specificities of plant transcription factors and their potential to define target genes. Proc Natl Acad Sci U S A. 111(6): 2367–2372. doi: 10.1073/pnas.1316278111 PMCID: PMC3926073

Gierson, D., Kader A.A. (1986). The Tomato Crop. Cap. 6. Fruit ripening and quality pp 241-280. DOI 10.1007/978-94-009-3137-4_6

Gupta, S., Stamatoyannopoulos, J., Bailey, T.L. and Noble, W.S. (2007). Quantifying similarity between motifs. Genome biology, 8(2), R24. doi:10.1186/gb-2007-8-2-r24

Klug, W.S. and Cummings, M.R. (2003) Concepts of Genetics. Prentice Hall, Upper Saddle River

Lin, I. (2012). Discovering Transcription Factor Binding Motif Sequences. Bioc218 Final Report.

Mathelier, A., Fornes, O., Arenillas, D.J., Chen, C., Denay, G., Lee, J., Shi, W., Shyr, C., Tan, G., Worsley-Hunt, R., Zhang, A.W., Parcy, F., Lenhard, B., Sandelin, A. and Wasserman, W.W. (2015). JASPAR 2016: a major expansion and update of the open-access database of transcription factor binding profiles. Nucl. Acids Res. 44 (D1): D110-D115.doi: 10.1093/nar/gkv1176

Poptsova, M.S., (2014). Genome analysis. In Current procedures and applications. Norfolk, UK.: Caister Academic Press.

Portales-Casamar, E., Thongjuea, S., Kwon, A., Arenillas, D., Zhao, X., Valen, E., Yusuf, D., Lenhard, B., Wasserman, W. and Sandelin, A. (2009). JASPAR 2010: the greatly expanded open-access database of transcription factor binding profiles. Nucleic Acids Research, vol. 38, issue SUPPL.1 , D105–D110. doi:10.1093/nar/gkp950

Tomato T, and Consortium G (2012). The tomato genome sequence provides insights into fleshy fruit evolution. Nature, 485, 635–41. doi:10.1038/nature11119

Zambelli, F., Pesole, G. and Pavesi, G. (2013). Motif discovery and transcription factor binding sites before and after the next-generation sequencing era. *Briefings in bioinformatics*, *14*(2), 225–37. doi:10.1093/bib/bbs016

Cock, P.A., Antao, T., Chang, J.T., Bradman, B.A., Cox, C.J., Dalke, A., Friedberg, I., Hamelryck, T., Kauff, F., Wilczynski, B. and de Hoon, M.J.L. (2009). Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25, 1422-1423

Gismondi, M. (2016). Estudio in silico de la expresión génica relativa a factores protectores frente al daño por frío en duraznos, Trabajo Final presentado para optar al grado académico de Especialista en Bioinformática, 68 páginas. Disponible en Biblioteca de la Facultad de Ciencias Agrarias UNR.

Suresh, B.V., Roy, R., Sahu, K., Misra, G., Chattopadhyay, D. (2014). Tomato Genomic Resources Database: An Integrated Repository of Useful Tomato Genomic Information for Basic and Applied Research. PLoS ONE 9(1): e86387. doi:10.1371/journal.pone.0086387