Universidad Autónoma de Madrid
Escuela politécnica superior

Máster en Bioinformática y Biología Computacional

Trabajo Fin de Máster

DESARROLLO DE UN WORKFLOW PARA EL ANÁLISIS GENÉTICO DE *CAMPYLOBACTER JEJUNI*

Autor: José Antonio Barbero Aparicio
Directores: José Francisco Díez Pastor y Beatriz Melero Gil
Ponente: Alberto Suárez González

Febrero 2019

DESARROLLO DE UN WORKFLOW PARA EL ANÁLISIS GENÉTICO DE *CAMPYLOBACTER JEJUNI*

Autor: José Antonio Barbero Aparicio
Directores: José Francisco Díez Pastor y Beatriz Melero Gil
Ponente: Alberto Suárez González

Dpto. de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Febrero 2019

Resumen

Resumen

Palabras Clave

Abstract

Keywords

Agradecimientos

Índice general

Índice de Figuras

Índice de Tablas

1 Introducción

1.1. Motivación del proyecto

Campylobacter jejuni es una bacteria Gram negativa que, a pesar de tener unas condiciones complicadas de crecimiento [1], es la zoonosis bacteriana que produce un mayor número de intoxicaciones alimentarias en los países tanto desarrollados como en vías de desarrollo. Por ejemplo, en la EU en el año 2016 se declararon del orden de 250.000 casos comprobados [2]. El coste debido a la campilobacteriosis se estima en la EU en torno a 2,4 billones de euros anuales. La fuente de contaminación más habitual es el consumo de carne de pollo poco cocinada [3]. El grupo de investigación Tecnofood[1] lleva varios años investigando sobre las fuentes de contaminación de este microorganismo a lo largo de la cadena alimentaria [1, 3, 4]. En la actualidad se dispone de una colección de Campylobacter spp. de alrededor de 2000 cepas. Con el fin de obtener una información más precisa sobre la persistencia de este microorganismo a lo largo de la cadena alimentaria, se han aislado varios genotipos persistentes en el matadero. De estos se han secuenciado con un equipo MiSeq (Illumina) 45 de ellas.

El proyecto consiste en diseñar un workflow que permita, a partir de los datos obtenidos en formato fastq proporcionados por el equipo, conseguir realizar las fases de trimming y evaluación de la calidad de las secuencias obtenidas, obtención de contigs, assembling y anotación, para poder tener la información de la secuencia de genes del genoma completo [5, 6, 7] de las cepas de *Campylobacter jejuni* secuenciadas. En la actualidad, existen varios programas desarrollados por varios grupos de investigación internacional que realizan las funciones demandadas. Se trata de buscar la solución más eficaz y fácil de implementar y que dé los mejores resultados, por lo que habrá que comparar diferentes programas y estrategias. Adicionalmente, se requiere incorporar en este workflow, o en análisis paralelos [8], la posibilidad de detectar insertos de origen viral y/o plásmidos en el genoma y herramientas que permitan la comparación rápida de los genomas de las distintas cepas aisladas, algunas de ellas pertenecientes a cepas altamente clonales. Esta herramienta se ha demandado por parte de un grupo sin conocimientos informáticos, por lo que se requiere desarrollar un entorno de fácil uso por su parte.

El proyecto plantea una colaboración entre los grupos ADMIRABLE [2] y TECNOFOOD de la Universidad de Burgos. Especializados en informática y ciencia y tecnología de los alimentos

[1] `https://www.ubu.es/tecnologia-de-los-alimentos-tecnofood`

[2] `https://www.ubu.es/advanced-data-mining-research-and-bioinformatics-learning-admirable`

Tarea 1 - Desarrollo de la imagen *Docker*	
Tarea 1.1 - Adaptar la imagen previa orientada a *Galaxy*	50 horas
Tarea 1.2 - Permitir desplegar la imagen en un servidor	10 horas
Tarea 2 - Desarrollo del workflow en *Galaxy*	
Tarea 2.1 - Selección de las herramientas	30 horas
Tarea 2.2 - Configuración completa del workflow	100 horas
Tarea 3 - Desarrollo del modo de uso simplificado	
	60 horas
Tarea 4 - Desarrollo del sistema de tratamiento de datos de salida	
	50 horas

Cuadro 1.1: Estimación en horas del plan de trabajo

respectivamente. Dada esta combinación de disciplinas, el proyecto se encuentra en el marco de los trabajos considerados dentro del campo de la bioinformática.

1.2. Objetivos y enfoque

El objetivo principal del proyecto es el desarrollo del workflow que permita el análisis de las cepas de *Campylobacter jejuni*, para lo que se utilizarán *Galaxy* [9] y las herramientas disponibles en la «tool shed» (conjunto de herramientas que ofrece *Galaxy* para su instalación) para cada paso. *Galaxy* es una herramienta que permite análisis computacionales de datos biológicos. El segundo objetivo se centra en crear una herramienta de utilización simplificada del workflow, en la que los pasos para su ejecución se configuren automáticamente, y que se llevará a cabo utilizando *Python* y la API de *Galaxy* a través de Bioblend [10]. Además, para facilitar el despliegue y ejecución de la herramienta desarrollada, se va a crear un contenedor *Docker*. Por lo tanto, el siguiente objetivo parcial será desarrollar la imagen *Docker* que sirva de base. Finalmente, se desea tener alguna forma sencilla de tratar los datos de salida del workflow, por lo que dentro de la interfaz gráfica se incluirán herramientas con las que gestionar toda la información resultante en forma de gráficos, tablas tipo hoja de cálculo, formatos pdf, etc.

Objetivos del proyecto:

- Desarrollar el workflow necesario para el análisis de las cepas en *Galaxy*
- Crear un sistema *Docker* sobre el que desplegar el proyecto
- Desarrollar una interfaz gráfica para simplificar la utilización de la aplicación
- Añadir un sistema de gestión sencilla de los datos de salida.

1.3. Metodología y plan de trabajo

1.3.1. Metodología

La metodología utilizada en el desarrollo del proyecto, dada su cercanía en su estructura a un proyecto de software tradicional, será de tipo ágil, basada en reuniones en cada sprint. La carga de trabajo, como aproximación a falta de conocer ciertos requisitos que puedan surgir durante el desarrollo, se dividirá en la estructura definida en la estimación en horas del plan de trabajo.

1.3.2. Plan de Trabajo

Sprint 1 (18/9/2018 - 3/10/2018)

El primer sprint ha estado centrado tanto en definir con más exactitud la dirección del proyecto como en un primer acercamiento a las principales herramientas con las que va a desarrollarse.

Tras unos primeros pasos con *Galaxy* [11] y *Docker* [12], se ha tomado como referencia el trabajo Bioinfworkflow de Sergio Chico [13] como base para la imagen *Docker* del proyecto. Dado que el proyecto de Github daba algunos problemas en la instalación, se ha desarrollado un script propio que produce los mismos resultados.

Una vez se ha tenido disponible la imagen de *Docker*, el sprint se ha centrado en algunos aspectos importantes para partes futuras del desarrollo. Entre ellos destaca la investigación acerca del formato de los workflows de *Galaxy* (.ga) ya que en un futuro será necesario generar este tipo de ficheros para introducirlos en *Galaxy*. También resulta relevante la investigación acerca de las posibilidades que ofrece la API de *Galaxy* [11] y su utilidad en Bioblend [14], que nos facilitan la opción de utilizar *Galaxy* sin necesidad de hacerlo a través de su interfaz.

Sprint 2 (4/10/2018 - 17/10/2018)

La primera semana de este sprint ha estado dirigida a lograr una imagen *Docker* de *Galaxy* que contenga un set de herramientas básicas para formar un primer workflow. Se han valorado varias opciones de instalación en las que se han utilizado tanto la imagen básica de *Galaxy* [15] como la imagen de Bioinfworkflow [13]. Finalmente, se ha optado por utilizar Bioinforworkflow ya que parte de las herramientas necesarias ya estaban incluidas. Para realizar esta tarea se ha creado un nuevo fichero Dockerfile así como el listado de herramientas necesarias para su instalación.

Sprint 3 (18/10/2018 - 31/10/2018)

El sprint ha estado centrado en la correcta ejecución del workflow con las herramientas iniciales desde *Galaxy*. Durante el proceso de configuración, han surgido varias complicaciones que han impedido terminar el workflow completo en este sprint. En un principio, han surgido problemas con el filtrado de calidad utilizando Prinseq. Este problema no ha llegado a ser resuelto en este sprint a falta de tratar el tema con el grupo de Tecnofood. A continuación, se encontraron ciertos problemas con el formato de salida de la herramienta Prokka. A pesar de que la salida está marcada como formato gff3, un parámetro interno lo etiquetaba como gff. Esto impedía que la salida de Prokka pudiese ser utilizada como entrada en las herramientas siguientes.

Dados estos errores, se decidió trabajar en paralelo con la API de *Galaxy* desde *Python*, para intentar ejecutar tanto las herramientas como el workflow de una manera menos restringida. Finalmente, se ha llevado a cabo el desarrollo necesario para subir los ficheros a un historial y ejecutar cada una de las herramientas del workflow desde *Python*.

Sprint 4 (1/11/2018 - 14/11/2018)

La prioridad en este punto se ha centrado en completar el workflow desde *Galaxy*. Han surgido varios problemas en esta tarea. La primera es un bug en Mac por el cual los archivos eliminados dentro de *Docker* no se eliminan del todo y quedan fijados en un fichero residual. Quizá esto pueda deberse a que OSX no soporta de manera nativa la virtualización a nivel de sistema operativo, sino que se basa en Hyperkit para crear una capa de virtualización. Esto implica que

cada cierto tiempo hay que eliminar la imagen completa de *Docker* para poder liberar espacio, dado el gran tamaño de los archivos con los que se trabaja. Debido a ello, la tarea de completar el workflow se ha visto retrasada. Además, la ejecución de la herramienta Roary a través de *Galaxy* ejecuta sin errores pero no devuelve ningún resultado, lo que ha impedido continuar con la parte final del workflow.

Además de la tarea ya comentada, en este sprint se ha trabajado en el acceso al contenedor *Docker* desde otro ordenador en una red local. Con el objetivo de desplegar el servicio en un servidor.

También se ha estudiado la posibilidad de desarrollar una interfaz gráfica, realizando unas pruebas en las que simplemente se muestra alguna información extraída de la API de *Galaxy* en etiquetas creadas con PyQt.

Sprint 5 (15/11/2018 - 28/11/2018)

Al igual que en los sprints anteriores, gran parte de la carga de trabajo se ha centrado en resolver problemas en la ejecución del workflow a través de *Galaxy*. Se han realizado numerosas ejecuciones para comprobar si las salidas de cada herramienta eran las correctas. Esto ha servido para concluir que, al parecer, un fallo en la herramienta Roary incluida en *Galaxy*, impide que los ficheros retornados tengan contenido. Esto se ha comprobado a través de la ejecución de Roary standalone con los mismos datos de entrada y los mismos parámetros, obteniendo de esta manera los ficheros correctos.

Para ahorrar en tiempos de ejecución se han utilizado los ficheros fasta ya generados previamente, no los generados con la herramienta Spades de nuestro propio workflow. En el próximo sprint se tratará de integrar la parte previa a los pasos que ya son correctos.

También se ha añadido un fichero .gitignore para evitar la existencia de ficheros irrelevantes en el repositorio.

Parte del trabajo de este sprint se ha destinado a la redacción de la propuesta de proyecto y de la introducción a la documentación del mismo.

Sprint 6 (29/11/2018 - 12/12/2018)

El trabajo de este sprint se ha centrado en fragmentar el proceso del workflow lo máximo posible para detectar dónde se están produciendo los errores. Inicialmente se ha acortado el workflow hasta el paso de ensamblaje con Spades, ya que los problemas surgían en este punto. Posteriormente se ha ejecutado el workflow individualmente en lugar de por colecciones. De esta manera, se ha detectado que el problema se esta dando en la ejecución de Spades con dos secuencias concretas: 590 y 443. Tras investigar probando varias ejecuciones con diferentes parámetros, se ha llegado a la conclusión de que no era un fallo de configuración de la herramienta, sino de hardware. Al ceder a *Docker* una cantidad mayor de memoria RAM (8 Gb), el problema se ha solucionado. A continuación, se ha pasado a ejecutar de nuevo por colecciones hasta el paso de Spades, para comprobar si con este cambio ha sido suficiente para que la ejecución sea correcta con este formato.

También se ha realizado una modificación del fichero .gitignore para mantener los archivos .pdf generados por LaTeX.

Sprint 7 (13/12/2018 - 26/12/2018)

La tarea principal de este sprint ha sido la creación de una herramienta Roary que poder integrar en *Galaxy*. Se valoró la opción de crear una nueva imagen *Galaxy* instalando la herramienta desde la creación inicial de *Docker*, pero finalmente se ha optado por subir esta versión de Roary al *tool shed* de Galaxy.

A continuación, se han realizado varias comprobaciones del funcionamiento de la integración de esta herramienta, con resultados positivos. Sin embargo, al ejecutar el workflow completo, parecen surgir de nuevo errores en la parte de Prokka, provocados por la ejecución previa de Spades.

También se ha añadido el apartado de la Introducción de la documentación.

Sprint 8 (27/12/2018 - 9/1/2019)

El trabajo en este sprint se ha enfocado a completar definitivamente la ejecución del workflow solucionando los errores existentes. La máquina virtual con la que se realizaban estas pruebas disponía de 4 GB de RAM, insuficiente para la ejecución con este conjunto de datos. Esto causaba algunos errores poco descriptivos al ejecutarlo. El equipo de pruebas se ha aumentado a 32 GB de RAM, cediendo 16 GB a la máquina virtual, solucionando así estos errores.

A continuación, se ha desarrollado un script *Python* a partir del cual realizar todas las tareas necesarias para ejecutar el workflow, utiizando la API de *Galaxy*.

Se ha invertido el poco tiempo restante del sprint en el desarrollo de la documentación, definiendo la estructura de los apartados de «estado del arte» y «sistema, diseño y desarrollo».

2

Análisis genético de *Campylobacter jejuni*. Estado del arte

2.1. Introducción

2.2. *Campylobacter jejuni*

2.3. Técnicas en bioinformática

2.3.1. Control de calidad

2.3.2. Ensamblado

2.3.3. Anotación

2.3.4. Análisis pangenómico

2.3.5. Workflows en bacterias / Workflows en general

2.4. Herramientas en informática

2.4.1. Virtualización

2.4.2. Plataforma de soporte del workflow

2.4.3. Herramientas GUI

3

Sistema, diseño y desarrollo

3.1. Estructura general del sistema

El sistema desarrollado se ha basado en *Docker* para evitar problemas de configuración y portabilidad, es decir, podrá ser desplegado en cualquier sistema que soporte esta tecnología. La estructura está basada en varias capas, todas ellas sobre el sistema anfitrión que elijamos, pudiendo adaptarse incluso a un servidor.

- La primera capa de esta estructura es el propio sistema anfitrión que, normalmente, será el equipo de utilice el usuario regularmente. Su única función es la de servir de base para la ejecución de la imagen *Docker*.
- Inmediatamente por encima del sistema anfitrión se encuentra la imagen *Docker*, encargada de facilitar la configuración y las dependencias necesarias para la ejecución de *Galaxy*.
- La imagen *Docker* incluye el despliegue de *Galaxy* sobre el puerto 8080. Por lo que podemos considerar como otra capa del sistema a la propia ejecución de *Galaxy*.
- El punto principal del trabajo se encuentra en el apartado de workflows dentro de *Galaxy*. En él se localiza la secuencia de herramientas que conforma el núcleo del proyecto.
- Exteriormente a *Galaxy* y *Docker*, se han desarrollado dos herramientas que interactúan con el workflow. La primera es un script *Python* que se encarga de toda la configuración y ejecución del workflow, para facilitar su utilización a usuarios no experimentados en informática. La segunda herramienta se encarga de tratar los datos obtenidos de la ejecución para generar un formato más fácil de comprender y manipular por el usuario.

3.2. Desarrollo y configuración de *Docker*

La imagen *Docker* utilizada está basada, con algunas modificaciones, en la desarrollada por Sergio Chico en su Trabajo de Fin de Máster [13]. La cual, a su vez, hereda de una imagen *Docker Galaxy* estable desarrollada por Björn A. Grüning [15].

Utilizando como punto de partida la imagen de Sergio Chico, se han realizado una serie de adaptaciones para adecuarla al uso de nuestro workflow. La primera modificación realizada fue la instalación de las nuevas herramientas necesarias que no estaban disponibles en ese momento. Al conjunto de herramientas iniciales se añadieron *Prinseq*, *Spades*, *Roary*, *ABRicate* y *Prokka*.

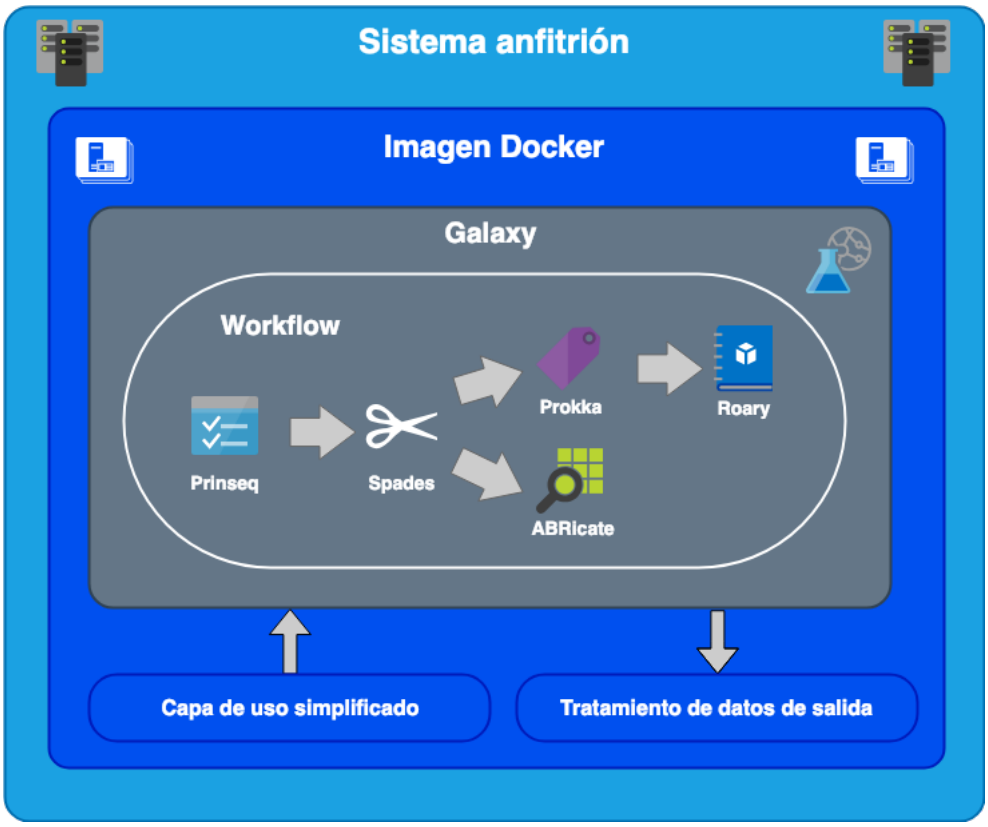

Figura 3.1: Estructura del sistema

Además, en las opciones de instalación de herramientas de *Galaxy* solo estaba disponible para la versión previa de *Roary* que producía, por algún tipo de bug, ficheros de salida vacíos. Por lo tanto, se creó una nueva herramienta a partir de la original con la nueva versión disponible para escritorio de *Roary*.

Una vez realizadas estas adaptaciones a la imagen, se ha dejado disponible en Dockerhub bajo el nombre *jbarberoaparicio/workflowmanager*[1].

Además de esto, se han desarrollado varios scripts simples para facilitar el uso de *Docker*. El primero de ellos se encarga del montaje completo de la imagen a partir del Dockerfile, en caso de que quisieran realizarse modificaciones en un futuro. El segundo se encarga del arranque de la imagen desplegándola a través de *Docker*. El tercero se desarrolló para resolver un bug en OSX por el cual los ficheros eliminados dentro de la imagen no se eliminaban correctamente. Esto provocaba que un fichero tipo *log* ocupando todo el espacio disponible en el disco. El script se encarga de eliminar esta información sin borrar las imágenes existentes.

3.3. Desarrollo y configuración del workflow en *Galaxy*

A lo largo del workflow, se realizarán varios análisis diferenciados de las secuencias. En algunos casos estos análisis servirán únicamente de entrada para el siguiente paso y en otros aportarán por sí mismos parte de la información deseada. Cabe destacar que se ha definido la entrada del workflow como un formato de dos colecciones, una para cada dirección de las lecturas de las secuencias.

[1] https://hub.docker.com/r/jbarberoaparicio/workflowmanager

3.3.1. *Prinseq*

Con el objetivo de realizar un filtrado de calidad de las secuencias, se ha utilizado *Prinseq* [16]. Esta herramienta, desarrollada en *Perl*, ofrece la posibilidad de generar un informe en el cual se indican diferentes estadísticas sobre la calidad de cierta secuencia. Además, utiliza esa información para realizar el filtrado que le indiquemos, pudiendo ser en función de la longitud de las lecturas o de su calidad.

Como formato de entrada, introduciremos en *Prinseq* un fichero fastq, en nuestro caso proveniente de las secuenciaciones de diferentes cepas de *Campylobacter jejuni* utilizando un equipo Illumina MiSeq. Como salida, obtendremos un fichero fastq en el que se habrán eliminado todos los fragmentos que no hayan cumplido las condiciones que se hayan fijado en los parámetros de ejecución. En nuestro caso, se han eliminado las secuencias de longitud menor que 50 y de calidad menor que 15. El resto de parámetros se han mantenido por defecto a excepción de la indicación de que se trata de una secuencia «paired-end», necesaria para la ejecución de la manera en la que deseamos llevarla a cabo.

Prinseq ha sido una de las herramientas que menos complicaciones han presentado a lo largo del proyecto, devolviendo errores comprensibles cuando las ejecuciones no eran correctas y ofreciendo toda la información necesaria para resolverlo. A pesar de ello, la versión de *Prinseq* disponible en *Galaxy* tiene la desventaja de no generar el informe con todas las estadísticas resultantes que sí genera su versión web.

3.3.2. *SPAdes*

SPAdes [17] ha sido la utilidad seleccionada para la fase de ensamblado. Se trata de un conjunto de herramientas que facilitan un modo de recoger todas las lecturas validadas en el proceso de filtrado y encadenarlas de la manera en la que corresponden en la secuencia. Fue desarrollado enfocándose a genomas pequeños, como bacterias y hongos. Por lo tanto, se adapta bien al caso de uso que estamos tratando.

Para su ejecución, se debe indicar a *SPAdes* la estructura que estamos siguiendo en el workflow, introduciendo ficheros separados en las dos direcciones en forma de colecciones. Estos ficheros provendrán de la salida de la ejecución de *Prinseq*, tras realizar el filtrado de calidad. El resto de parámetros de la ejecución se han mantenido por defecto, pudiendo destacar que se usan tamnaños de k-mers de 21, 33 y 55 por ser los recomendados. También se le ha indicado que las bibliotecas a utilizar sean «paired-end».

Dado el gran tamaño de las lecturas de los casos en los que el workflow ha sido probado, *SPAdes* ha resultado problemático por tener que almacenar en RAM todas las lecturas al mismo tiempo. Esto ha implicado que, para las muestras de prueba, se haya requerido de 16 GB de memoria RAM para poder ejecutarlo. Además, los mensajes de error de la versión *Galaxy* de *SPAdes* no dan demasiada información en muchos casos. En algunos otros, relacionados con la capacidad de memoria, no se producía ningún mensaje de error, sino que se daba por correcta la ejecución a pesar de que los ficheros retornados estaban vacíos. A pesar de estos problemas de desarrollo, *SPAdes* ha terminado cumpliendo su función correctamente tras identificar los errores del proceso.

3.3.3. *ABRicate*

Uno de los principales intereses al comenzar el proyecto era el análisis de resistencia a antibióticos de las secuencias. *ABRicate* [18] es una herramienta desarrollada en *Perl* que permite la localización de genes vinculados a resistencia a antibióticos o relacionados con virus realizando

un cribado masivo de los contigs introducidos. Para ello utiliza las bases de datos de Resfinder, CARD, ARG-ANNOT, NCBI BARRGD, NCBI, EcOH, PlasmidFinder, Ecoli_VF y VFDB, En nuestro caso, ese input serán los contigs obtenidos por *SPAdes* para cada una de las secuencias. Esta herramienta forma uno de los finales del workflow, es decir, su salida no sirve como entrada a ninguna otra herramienta sino que los datos que se obtienen de ella tienen su utilidad propia.

La configuración de ABRicate es muy sencilla y ofrece pocas posibilidades de variación, por lo que no ha sido necesario modificar ningún parámetro de la herramienta. La ejecución por defecto no ha presentado problemas.

Una de las limitaciones de *ABRicate* es que no admite lecturas tipo fastq, solamente contigs. En nuestro caso los contigs provienen de la salida de *SPAdes*, por lo que es una herramienta adecuada.

3.3.4. *Prokka*

Para la fase de anotación se ha decidido utilizar *Prokka* [19], una herramienta desarrollada en *Perl* destinada específicamente a genomas procariotas y que destaca por su rapidez. El proceso de anotación de *Prokka* se divide en dos fases. En la primera, se utiliza *Prodigal* para la identificación de regiones codificadoras de proteínas. En la segunda, se compara con varias bases de datos para reconocer, por similitud, cuál es la proteína de la que se trata.

La configuración de *Prokka* en *Galaxy* no ha resultado complicada a pesar de que los parámetros son numerosos. El único problema encontrado en la configuración ha sido a raíz del formato GFF3. A pesar de que se indique que los ficheros de salida siguen esa estructura, realmente no es así, sino que internamente son formato GFF. Para solucionar este inconveniente se ha añadido una fase nueva en el workflow, en la que una herramienta simple de concatenación une la salida GFF con la propia secuencia del genoma.

3.3.5. *Roary*

Uno de los objetivos principales del proyecto era la realización de un análisis pangenómico. Utilizando como entrada las anotaciones provenientes de *Prokka* de todas las muestras, *Roary* [20] se encarga de calcular el pangenoma. Ha resultado ser una herramienta realmente rápida, con la que en unos minutos se obtienen los resultados.

Roary está desarrollado para recibir ficheros GFF3 desde *Prokka*, por lo que no debería haber problemas de compatibilidad (una vez solucionado el error de formato comentado en el punto anterior). El problema con la configuración de *Roary* dentro del workflow es que, si se instala desde el repositorio oficial en el *Tool Shed* de *Galaxy*, no se obtiene ningún tipo de contenido en los ficheros de salida. Estudiando este problema, se llegó a la conclusión de que se podría deber a que la versión disponible en este repositorio no es la versión final publicada para escritorio. Por lo tanto, se creó una herramienta *Galaxy* nueva utilizando la versión *standalone* de *Roary*. Una vez disponible esta herramienta en *Galaxy* y manteniendo la configuración establecida para la versión anterior, los ficheros fueron obtenidos correctamente.

Roary recibe todos los ficheros tratados por el workflow (a excepción de la salida independiente de ABRicate) y sirve como herramienta final. Los datos obtenidos por *Roary* no son enviados a ninguna otra herramienta dentro de *Galaxy* y finalizan el workflow.

3.4. Desarrollo de la capa externa con *Python*

A pesar de que el workflow completo ha sido desarrollado dentro de *Galaxy*, se han añadido algunas funcionalidades extra utilizando *Python*.

3.4.1. Capa de uso simplificado

Dado que el equipo al que está destinada la herramienta no pertenece al campo de la informática, se solicitó una herramienta que fuese sencilla de utilizar. Por ello, se ha creado un script que realiza todos los pasos necesarios para ejecutar el workflow simplemente con ejecutarlo, sin ser necesario ningún tipo de configuración. En caso de que el usuario tuviese conocimientos técnicos, seguirá teniendo la posibilidad de utilizar la herramienta sin esta script.

Para ello, se ha utilizado la API de *Galaxy*, parte de la librería *BioBlend*[2]. Esta API nos permite controlar todos los aspectos de una instancia *Galaxy* desde código *Python*.

Desde el script se crea una sesión en la instancia de *Galaxy* a partir de la cual se realizan todas las tareas. Una vez conectado, el script carga el archivo que almacena el workflow que va a ejecutar. Después, para mantener los datos aislados e identificados, crea historiales nuevos para los ficheros de entrada y los de salida con un nombre único a partir de la fecha y hora de creación. A continuación carga los datos brutos de inicio, lo que, dependiendo de su tamaño, puede llevar unos minutos. Para que el workflow funcione correctamente, necesita recibir los datos en forma de colección, por lo que el script se encarga de crear una para cada conjunto de datos, es decir, lecturas en un sentido y en el inverso. El paso siguiente, la ejecución del workflow, es el fundamental y el más costoso en tiempo. Para controlar el estado de la tarea en cada momento, se va realizando un control cada pocos segundos que comprueba si las tareas siguen en proceso. Una vez finalizado, los datos de salida se mantienen en un historial propio, pero para poder utilizarlos fuera de *Galaxy*, el script se encarga de almacenarlos en el disco local del equipo en una carpeta con el nombre del historial.

3.4.2. Agrupación de resultados

Para facilitar el acceso a los resultados de todas las herramientas, se ha planteado la creación de un solo fichero que pueda abrirse con *Excel*, herramienta que resulta familiar al grupo que utilizará el workflow. Para ello, desde un script de *Python* se genera un solo fichero en el que cada hoja contiene los resultados de uno de los pasos del workflow.

[2]https://bioblend.readthedocs.io/en/latest/

4

Experimentos Realizados y Resultados

4.1. Bases de datos y protocolo

4.2. Sistemas de referencia

4.3. Escenarios de pruebas

4.4. Experimentos del sistema completo

5

Conclusiones y trabajo futuro

Glosario de acrónimos

- **IS**: Iris Subject
- **DCT**: Discrete Cosine Transform
- **WED**: Weighted Euclidean Distance

Bibliografía

[1] Lourdes García-Sánchez, Beatriz Melero, Isabel Jaime, Marja-Liisa Hänninen, Mirko Rossi, and Jordi Rovira. Campylobacter jejuni survival in a poultry processing plant environment. *Food Microbiology*, 65:185–192, aug 2017.

[2] Campylobacteriosis - Annual Epidemiological Report 2016 [2014 data].

[3] Lourdes García-Sánchez, Beatriz Melero, Ana Ma Diez, Isabel Jaime, and Jordi Rovira. Characterization of Campylobacter species in Spanish retail from different fresh chicken products and their antimicrobial resistance. *Food Microbiology*, 76:457–465, dec 2018.

[4] Beatriz Melero, Pekka Juntunen, Marja-Liisa Hänninen, Isabel Jaime, and Jordi Rovira. Tracing Campylobacter jejuni strains along the poultry meat production chain from farm to retail by pulsed-field gel electrophoresis, and the antimicrobial resistance of isolates. *Food Microbiology*, 32(1):124–128, oct 2012.

[5] Clifford G. Clark, Chrystal Berry, Matthew Walker, Aaron Petkau, Dillon O. R. Barker, Cai Guan, Aleisha Reimer, and Eduardo N. Taboada. Genomic insights from whole genome sequencing of four clonal outbreak Campylobacter jejuni assessed within the global C. jejuni population. *BMC Genomics*, 17(1):990, dec 2016.

[6] Ann-Katrin Llarena, Eduardo Taboada, and Mirko Rossi. Whole-Genome Sequencing in Epidemiology of Campylobacter jejuni Infections. *Journal of clinical microbiology*, 55(5):1269–1275, may 2017.

[7] S. Zhao, G. H. Tyson, Y. Chen, C. Li, S. Mukherjee, S. Young, C. Lam, J. P. Folster, J. M. Whichard, and P. F. McDermott. Whole-Genome Sequencing Analysis Accurately Predicts Antimicrobial Resistance Phenotypes in Campylobacter spp. *Appl. Environ. Microbiol.*, 82(2):459–466, jan 2016.

[8] C. P. A. Skarp, O. Akinrinade, A. J. E. Nilsson, P. Ellström, S. Myllykangas, and H. Rautelin. Comparative genomics and genome biology of invasive Campylobacter jejuni. *Scientific Reports*, 5(1):17300, dec 2015.

[9] Enis Afgan, Dannon Baker, Bérénice Batut, Marius Van Den Beek, Dave Bouvier, Martin Čech, John Chilton, Dave Clements, Nate Coraor, Björn A Grüning, et al. The galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update. *Nucleic acids research*, 46(W1):W537–W544, 2018.

[10] Clare Sloggett, Nuwan Goonasekera, and Enis Afgan. BioBlend: automating pipeline analyses within Galaxy and CloudMan. *Bioinformatics (Oxford, England)*, 29(13):1685–6, jul 2013.

[11] Galaxy. `https://usegalaxy.org/`.

[12] Docker - Build, Ship, and Run Any App, Anywhere. `https://www.docker.com/`.

[13] Sergio Chico. :whale: Docker with Galaxy for Bioinformatic Bacterial Sequencing Workflows: Serux/docker-galaxy-BioInfWorkflow. `https://github.com/Serux/docker-galaxy-BioInfWorkflow`, June 2018. original-date: 2018-05-17T01:10:05Z.

[14] Galaxy API. `https://galaxyproject.org/develop/api/`.

[15] Björn Grüning. :whale::bar_chart::books: Docker Images tracking the stable Galaxy releases.: bgruening/docker-galaxy-stable. https://github.com/bgruening/docker-galaxy-stable, October 2018. original-date: 2014-08-12T13:26:14Z.

[16] Robert Schmieder and Robert Edwards. Quality control and preprocessing of metagenomic datasets. *Bioinformatics (Oxford, England)*, 27(6):863–864, March 2011. PMID: 21278185.

[17] Sergey Nurk, Anton Bankevich, Dmitry Antipov, Alexey Gurevich, Anton Korobeynikov, Alla Lapidus, Andrey Prjibelsky, Alexey Pyshkin, Alexander Sirotkin, Yakov Sirotkin, Ramunas Stepanauskas, Jeffrey McLean, Roger Lasken, Scott R. Clingenpeel, Tanja Woyke, Glenn Tesler, Max A. Alekseyev, and Pavel A. Pevzner. Assembling Genomes and Mini-metagenomes from Highly Chimeric Reads. pages 158–170. Springer, Berlin, Heidelberg, 2013.

[18] Torsten Seemann. :mag_right: :pill: Mass screening of contigs for antimicrobial and virulence genes: tseemann/abricate, January 2019. original-date: 2014-07-17T00:59:35Z.

[19] T. Seemann. Prokka: rapid prokaryotic genome annotation. *Bioinformatics*, 30(14):2068–2069, jul 2014.

[20] Andrew J. Page, Carla A. Cummins, Martin Hunt, Vanessa K. Wong, Sandra Reuter, Matthew T.G. Holden, Maria Fookes, Daniel Falush, Jacqueline A. Keane, and Julian Parkhill. Roary: rapid large-scale prokaryote pan genome analysis. *Bioinformatics*, 31(22):3691–3693, nov 2015.

A

Manual de utilización

B

Manual del programador