
Ensamblaje y Anotación de Genomas Bacterianos

Manual de Bioinformática

Héctor Fabio Espitia-Navarro
Alvaro Pazos M.



Universidad de **Nariño**
FUNDADA EN 1904



Universidad de **Nariño**
ACREDITADA DE ALTA CALIDAD
RESOLUCIÓN MEN 10567 - MAYO 23 DE 2017



CESUN
CENTRO DE ESTUDIOS EN SALUD
Universidad de **Nariño**



Universidad de **Nariño**
FUNDADA EN 1904



Universidad de **Nariño**
ACREDITADA DE ALTA CALIDAD
RESOLUCIÓN MEN 10567 - MAYO 23 DE 2017



C E S U N
CENTRO DE ESTUDIOS EN SALUD
Universidad de **Nariño**

ENSAMBLAJE Y ANOTACIÓN DE GENOMAS BACTERIANOS

Manual de Bioinformática

ENSAMBLAJE Y ANOTACIÓN DE GENOMAS BACTERIANOS

Manual de Bioinformática

Héctor Fabio Espitia-Navarro Alvaro Pazos M.



Universidad de **Nariño**
FUNDADA EN 1904



Universidad de **Nariño**
ACREDITADA DE ALTA CALIDAD
RESOLUCIÓN MEN 10567 - MAYO 23 DE 2017



CESUN
CENTRO DE ESTUDIOS EN SALUD
Universidad de **Nariño**

ENSAMBLAJE Y ANOTACIÓN DE GENOMAS BACTERIANOS: MANUAL DE BIOINFORMÁTICA

© Héctor Fabio Espitia-Navarro
Alvaro Pazos M.

© Universidad de Nariño

Diagramación y diseño: Héctor Fabio Espitia-Navarro

Fecha de publicación: 2023
San Juan de Pasto, Nariño, Colombia

Prohibida la reproducción total o parcial, por cualquier medio o con cualquier propósito,
sin la autorización escrita de sus Autores o de la Universidad de Nariño.

Tabla de contenidos

Bienvenid@	3
Por qué leer este manual	3
Prerrequisitos	4
Convenciones del manual	4
Tipografía	4
Bloques especiales	5
1. Preparación del entorno de trabajo	6
1.1. Estructura de directorios	6
1.2. Ambiente Conda e instalación de programas	7
1.2.1. Instalación Conda: mambaforge	7
1.2.2. Creación del ambiente Conda	10
1.2.3. Instalación de paquetes	11
2. Obtención de datos	13
2.1. Software requerido	13
2.2. Datos	13
2.3. Descarga	14
3. Control de calidad y limpieza de datos	17
3.1. Software requerido	17
3.2. Determinación de calidad	18
3.2.1. Reporte de calidad FastQC	19
3.2.2. Unificando los reportes de calidad	21
3.3. Limpieza	22
3.4. Verificación de calidad después de la limpieza	29
4. Ensamblaje	31
4.1. Software requerido	31
4.2. Ensamblaje de genomas	31
5. Control de calidad de ensamblajes	39
5.1. Software requerido	39

Tabla de contenidos

5.2. Determinación de calidad	39
5.2.1. Descarga del genoma de referencia	40
5.2.2. Ejecución de Quast	41
5.3. Evaluando la calidad	42
5.4. Corrección de ensamblajes	45
6. Anotación	47
6.1. Software requerido	47
6.2. Ejecución de Prokka	47
Bibliografía	52
Acerca de los autores	54
Héctor Fabio Espitia-Navarro	54
Alvaro Pazos Moncayo	55
Apéndices	57
A. FastQC help	57
A.1. Introduction	57
A.1.1. What is FastQC	57
A.2. Basic Operations	58
A.2.1. Opening a Sequence file	58
A.2.2. Evaluating Results	58
A.2.3. Saving a Report	59
A.3. Analysis Modules	59
A.3.1. Basic Statistics	59
A.3.2. Adapter Content	60
A.3.3. Kmer Content	62
A.3.4. Per Tile Sequence Quality	64
A.3.5. Per Base Sequence Quality	66
A.3.6. Per Sequence Quality Scores	68
A.3.7. Per Base Sequence Content	70
A.3.8. Per Sequence GC Content	73
A.3.9. Per Base N Content	74
A.3.10. Sequence Length Distribution	76
A.3.11. Duplicate Sequences	78
A.3.12. Overrepresented Sequences	81

Bienvenid@

Este es “Ensamblaje y Anotación de Genomas Bacterianos”, un manual de bioinformática para ensamblar y anotar genomas de bacterias a partir de datos de secuenciación de genoma completo (WGS, del inglés *Whole Genome Sequencing*) proveniente de aislamientos. El manual presenta el proceso más sencillo de ensamblaje y anotación de genomas, considerando los pasos básicos de bioinformática necesarios para lograrlo.

Por qué leer este manual

La secuenciación de genomas bacterianos completos actualmente se ha convertido en una herramienta bastante útil y poderosa en el ámbito de la microbiología y la epidemiología. Especialmente, la epidemiología molecular, se apoya cada vez más en los genomas obtenidos mediante las tecnologías de secuenciación de segunda generación (NGS, del inglés *Next Generation Sequencing*) para la vigilancia y seguimiento de brotes infecciosos causados por patógenos bacterianos. Los genomas obtenidos mediante NGS consisten en millones de secuencias cortas de ADN las cuales deben ser procesadas mediante bioinformática para extraer el conocimiento útil para epidemiología. Este proceso se llama ensamblaje de genomas, uno de los análisis básicos en el ámbito de la bioinformática.

Actualmente las habilidades en bioinformática más que ser una ventaja, representan una necesidad en el mundo tan competitivo de las ciencias biológicas, especialmente en la microbiología, la cual se apoya cada vez más en los análisis informáticos. Este manual provee el conocimiento de bioinformática básico para el ensamblaje y anotación de genomas bacterianos. El manual desarrolla el flujo de trabajo más básico: obtención de datos, control de calidad de los datos, ensamblaje, y anotación funcional. Quedan fuera de este documento procedimientos más avanzados como la corrección de ensamblajes, los cuales posiblemente sean incluidos aquí en un futuro.

Prerrequisitos

El manual está destinado para ser ejecutado en el sistema Operativo (SO) Linux. Específicamente, se usó la distribución Ubuntu Linux 20.04 para desarrollar este manual. Aunque no se requiere que sea un usuario Linux avanzado, será más fácil seguir el manual si está familiarizado con el uso de la línea de comandos en el SO Linux. Puede aprender más sobre Linux y la línea de comandos en el manual [“Introducción a Linux para Bioinformática”](#).

Convenciones del manual

Tipografía

Las siguientes convenciones tipográficas son usadas en este manual:

- *Cursiva*:
 - Palabras escritas en idiomas diferentes del español (p. ej. *prompt*, o *forward*)
 - Expresiones aritméticas (p. ej. $Q \geq 20$).
- Fuente monoespaciada:
 - Nombres de archivos (p. ej. `/ruta/al/archivo.txt`)
 - Nombres especiales (p. ej. ambientes conda: `base`)
 - Teclas (p. ej. `ENTER`)
 - Nombre de comandos (p. ej. `ls`).
 - Bloques de órdenes de línea de comandos y su salida. Por ejemplo:

```
ls -l /home
```

```
total 4
```

```
drwxr-xr-x 39 hector hector 4096 mar 28 12:37 hector
```

- **Hipervínculos (links)**: enlaces a partes de este documento (internos) o a otros sitios Web (externos).

Bloques especiales

Textos con información de especial atención se presentan en bloques (recuadros) especiales que se diferencian por su color. Existen cinco tipos de bloques especiales:

Nota

Este es un ejemplo de una nota.

Importante

Este es un texto que corresponde a una información importante.

Tip

Así se ven los consejos o trucos

Precaución

Este bloque denota una información de precaución

Advertencia

Este bloque denota una información de advertencia.

1. Preparación del entorno de trabajo

1.1. Estructura de directorios

Para que cada proyecto esté organizado en carpetas independientes, primero crearemos el directorio para los proyectos. Abra una terminal y ejecute las siguientes órdenes:

```
cd $HOME
mkdir proyectos
```

! Importante

Se asume que usted debe ejecutar en la terminal todos los comandos mostrados a partir de este punto.

Ahora, se creará la estructura para nuestro proyecto de ensamblaje dentro del directorio `proyectos`. En la estructura se contemplan subdirectorios de datos, calidad y resultados:

```
mkdir -p proyectos/ensamblaje/{datos, resultados, scripts}
mkdir -p proyectos/ensamblaje/datos/qc/fastqc
mkdir -p proyectos/ensamblaje/resultados/{00_datos_limpios,01_ensamblaje,02_ensamblaje_qc,03_annotacion}
mkdir -p
  ↳ proyectos/ensamblaje/resultados/00_datos_limpios/qc/{fastp, fastqc}
```

La estructura final del directorio del directorio del proyecto se puede ver con el comando `tree`:

```
tree -L 4 --charset HTML proyectos/ensamblaje
```

```
proyectos/ensamblaje
|-- datos
|   |-- qc
```

1. Preparación del entorno de trabajo

```
|         |-- fastqc
|-- resultados
|   |-- 00_datos_limpios
|   |   |-- qc
|   |       |-- fastp
|   |       |-- fastqc
|   |-- 01_ensamblaje
|   |-- 02_ensamblaje_qc
|   |-- 03_ anotacion
|-- scripts
```

12 directories, 0 files

1.2. Ambiente Conda e instalación de programas

Una manera eficiente y rápida para instalar todos los programas necesarios para el ensamblaje y anotación de los genomas, consiste en usar Conda¹, un software de libre distribución para administrar ambientes de software e instalación de programas.

1.2.1. Instalación Conda: mambaforge

Instalaremos una implementación liviana de Conda de la comunidad conda-forge² llamada miniforge³, que además viene con el gestor de paquetes `mamba`, el cual es más rápido y eficiente que `conda`.

Primero, descarguemos el instalador:

```
wget https://github.com/conda-forge/miniforge/releases/latest/download_
  ↪ /Mambaforge-Linux-x86_64.sh
```

i Nota

Toda la información acerca de miniforge y otros instaladores están disponibles en <https://github.com/conda-forge/miniforge>

¹Documentación de Conda (en inglés): <https://docs.conda.io/en/latest/>

²Conda-forge: <https://conda-forge.org/>

³Miniforge en GitHub: <https://github.com/conda-forge/miniforge>.

1. Preparación del entorno de trabajo

Ahora iniciemos el instalador:

```
# Adicionar permiso de ejecución
chmod +x Mambaforge-Linux-x86_64.sh

# Ejecutar el instalador
./Mambaforge-Linux-x86_64.sh
```

Al ejecutar el anterior comando, verá un mensaje de los términos de la licencia como este:

```
Welcome to Mambaforge 22.11.1-4

In order to continue the installation process, please review the license
agreement.
Please, press ENTER to continue
>>>
```

Presione la tecla **ENTER** varias veces hasta que el instalador solicite aceptar los términos de la licencia:

```
Do you accept the license terms? [yes|no]
[no] >>>
```

Responda **yes**

A continuación el instalador solicitará el lugar donde instalará Mambaforge. Puede confirmar la ubicación por defecto (en su directorio *home*: `~/mambaforge`) presionando **ENTER**:

```
Mambaforge will now be installed into this location:
/home/hector/mambaforge

- Press ENTER to confirm the location
- Press CTRL-C to abort the installation
- Or specify a different location below

[/home/hector/mambaforge] >>>
```

Una vez el instalador finaliza la copia de archivos, le preguntará si desea inicializar Mambaforge:

1. Preparación del entorno de trabajo

```
...
Transaction finished
installation finished.
Do you wish the installer to initialize Mambaforge
by running conda init? [yes|no]
[no] >>>
```

Responda `yes`. Después, verá un mensaje que indica los cambios realizados en los archivos de configuración para inicializar Mambaforge:

```
...
no change      /home/hector/mambaforge/bin/conda
no change      /home/hector/mambaforge/bin/conda-env
no change      /home/hector/mambaforge/bin/activate
no change      /home/hector/mambaforge/bin/deactivate
no change      /home/hector/mambaforge/etc/profile.d/conda.sh
no change      /home/hector/mambaforge/etc/fish/conf.d/conda.fish
no change      /home/hector/mambaforge/shell/condabin/Conda.ps1
no change      /home/hector/mambaforge/shell/condabin/conda-hook.ps1
no change      /home/hector/mambaforge/lib/python3.10/site-packages/xont
↳ rib/conda.xsh
no change      /home/hector/mambaforge/etc/profile.d/conda.csh
modified       /home/hector/.bashrc

==> For changes to take effect, close and re-open your current shell. <==

Added mamba to /home/hector/.bashrc

==> For changes to take effect, close and re-open your current shell. <==
```

Para que la instalación tome efecto, cierre y abra nuevamente la terminal.

Seguramente cuando abra nuevamente la terminal, notará que el *prompt* luce diferente; algo parecido a esto:

```
(base) hector@Ubuntu:~$
```

Esto es debido a que Conda usa el *prompt* como medio para informar el ambiente se encuentra activo (en uso) actualmente, mostrándolo entre paréntesis al inicio del prompt. En el ejemplo anterior aparece la palabra `base`; este es el nombre del ambiente “base” por defecto de Conda.

1. Preparación del entorno de trabajo

1.2.2. Creación del ambiente Conda

El siguiente paso es crear un ambiente Conda en donde se instalarán todas los programas requeridos. Un ambiente es un directorio que contiene una colección específica de paquetes (programas) de Conda (con una versión específica) que se han instalado.

Para crear el ambiente, ejecute las siguientes órdenes:

```
mamba create -n ensam

...
Looking for: []

Preparing transaction: done
Verifying transaction: done
Executing transaction: done

To activate this environment, use

    $ mamba activate ensam

To deactivate an active environment, use

    $ mamba deactivate
```

Como pudo ver, `mamba` despliega un mensaje de confirmación de creación del ambiente.

! Importante

Para poder usar el ambiente creado, siempre se debe activarlo primero:

```
mamba activate ambiente
```

donde `ambiente` es el nombre del ambiente creado.

Cuando ya no se necesita el ambiente, se debe desactivar:

```
mamba deactivate
```

1. Preparación del entorno de trabajo

1.2.3. Instalación de paquetes

Ahora instalaremos los paquetes requeridos. Primero active el ambiente:

```
mamba activate ensam
```

i Nota

Cuando se activa correctamente, podrá ver el nombre del ambiente recién activado en el *prompt*, en este caso `ensam` en lugar de `base`:

```
(ensam) hector@Ubuntu:~$
```

Ahora instale los paquetes ejecutando el siguiente comando:

```
mamba install -c bioconda csvtk fastp fastqc java-jdk multiqc \
ncbi-datasets-cli pigz prokka quast spades sra-tools wget
```

A continuación verá algunos mensajes y barras de progreso mientras mamba descarga los listados de paquetes y busca los paquetes que se ordenaron instalar. Al final mamba solicitará que confirme la instalación de los paquetes:

```
...
Summary:

Install: 290 packages

Total download: 1GB
```

```
Confirm changes: [Y/n]
```

Confirme escribiendo `Y` y después presione `ENTER`. De nuevo verá una serie de tareas completadas y barras de progreso mientras la instalación transcurre.

Aunque se ordenó instalar 11 paquetes, cada uno de ellos requiere múltiples dependencias, es decir, paquetes adicionales para funcionar. Aquí es donde se facilitan las cosas:

1. Preparación del entorno de trabajo

mamba (o conda) se encarga de revisar las dependencias de cada paquete, instalarlas y configurarlas automáticamente para que cada programa funcione adecuadamente.

Al finalizar la instalación verá un mensaje parecido a este:

```
...
Downloading and Extracting Packages

Preparing transaction: done
Verifying transaction: done
Executing transaction: \ The default QUASt package does not include:
* GRIDSS (needed for structural variants detection)
* SILVA 16S rRNA database (needed for reference genome detection in
  ↳ metagenomic datasets)
* BUSCO tools and databases (needed for searching BUSCO genes) -- works
  ↳ in Linux only!

To be able to use those, please run
  quast-download-gridss
  quast-download-silva
  quast-download-busco

done
```

Ahora todo el software requerido para el análisis de los datos está instalado.

2. Obtención de datos

2.1. Software requerido

Software	Versión	Descripción	Ref.
SRA Tools	3.0.3	Colección de herramientas para descargar datos desde la base de datos SRA ¹ (del inglés <i>Sequence Read Archive</i> ; Katz et al. (2022)) del NCBI (del inglés <i>National Center for Biotechnology Information</i>)	Repositorio GitHub

2.2. Datos

Descargaremos cuatro muestras de secuenciación de genoma completo o WGS (del inglés *Whole Genome Sequencing*) del organismo *Helicobacter pylori*, para ser ensambladas. Para descargar estas muestras necesitaremos los identificadores (número de acceso o *accession number* en inglés) en la base de datos SRA:

1. SRR22388518
2. SRR22388519
3. SRR18335438
4. SRR18335439

¹Sequence Read Archive: <https://www.ncbi.nlm.nih.gov/sra>

2.3. Descarga

! Importante

Recuerde siempre activar el ambiente `ensam` (si no está activado ya) antes de ejecutar los comandos de este manual:

```
mamba activate ensam
```

Para empezar a descargar las muestras, abra una terminal y ejecute las órdenes que se dan a continuación.

⚠ Advertencia

Siempre que vea comentarios (líneas empezando con `#`) en las cajas de comandos, no los digite en la terminal. Estos comentarios solo proveen información adicional a cerca de cada comando.

i Nota

- La descarga de cada muestra toma aproximadamente cinco minutos (o más), dependiendo de la velocidad de su conexión a Internet. No se preocupe si no ve ningún avance aparente después de ejecutar el comando de descarga (`fasterq-dump`).
- Si quiere ver el progreso de cada descarga, use la opción `-p` (o `--progress`) con `fasterq-dump`
- Puede ver otras opciones útiles de `fasterq-dump` consultando la ayuda: `fasterq-dump --help`

```
# Entrar al directorio para datos en el directorio del proyecto
cd $HOME/proyectos/ensamblaje/datos
```

```
# Descargar las muestras
fasterq-dump -S SRR22388518
```

```
spots read      : 870,940
reads read      : 1,741,880
reads written   : 1,741,880
```

2. Obtención de datos

Finalizada la descarga `fasterq-dump` imprime en pantalla el resumen de las secuencias descargadas y escritas en archivos (salida anterior) .

Ahora verifique los archivos descargados:

```
ls -lh
```

```
total 962M
-rw-rw-r-- 1 hector hector 481M mar  2 14:40 SRR22388518_1.fastq
-rw-rw-r-- 1 hector hector 481M mar  2 14:40 SRR22388518_2.fastq
```

Como puede observar, hay dos archivos descargados con la extensión `.fastq`. Veamos la estructura de nombre de los archivos descargados:

- `SRR22388518`: *accession number*.
- `1`: conjunto de secuencias, en este caso *forward*.
- `fastq`: extensión de archivo.

! Importante

Hay dos archivos dado que la muestra se secuenció con tecnología Illumina *paired-end*, en la cual resultan dos *reads* (secuencias), *forward* (`_1`) y *reverse* (`_2`), de secuenciar los extremos de una molécula de ADN. En la Figura 2.1 se detalla la ubicación relativa de los dos *reads* respecto del fragmento de ADN secuenciado. En el archivo `_1` están guardadas todas las secuencias *forward* y en el archivo `_2`, todas las secuencias *reverse*.

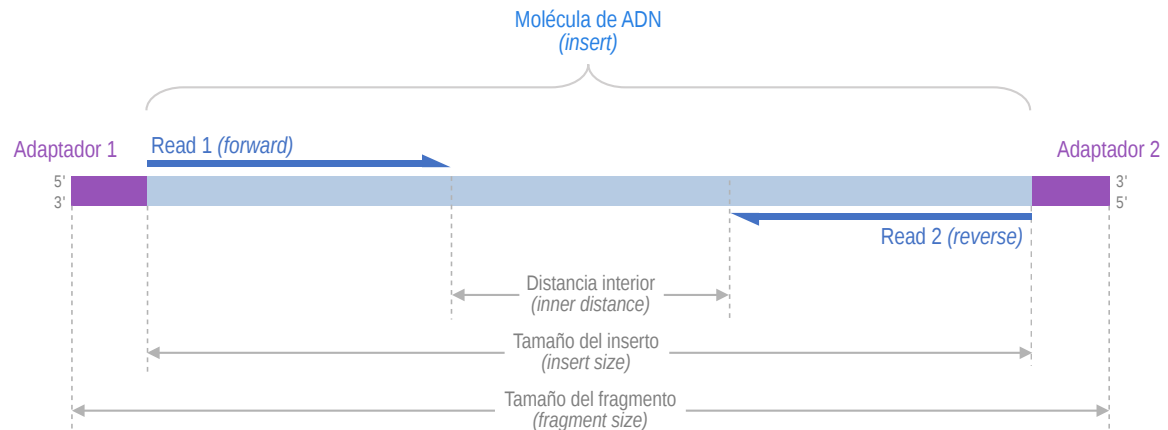


Figura 2.1.: Representación esquemática de la secuenciación *paired-end*.

2. Obtención de datos

El formato FASTQ² (Cock et al. 2010) es el estándar *de facto* para datos de secuenciación de segunda generación y es usado, entre otras, por la tecnología Illumina. Este formato almacena tanto las secuencias de ADN como la calidad de cada base secuenciada.

Ahora descarguemos el resto de las muestras:

```
fasterq-dump -S SRR22388519
```

```
fasterq-dump -S SRR18335438
```

```
fasterq-dump -S SRR18335439
```

Ahora, aunque no es necesario, es una buena práctica comprimir los archivos descargados para que ocupen menos espacio en disco:

```
gzip *.fastq
```

Como verá, ahora los archivos están comprimidos, por lo que tienen la extensión `.gz` (`.fastq.gz`):

```
ls
```

```
SRR18335438_1.fastq.gz  SRR18335439_2.fastq.gz  SRR22388519_1.fastq.gz  
SRR18335438_2.fastq.gz  SRR22388518_1.fastq.gz  SRR22388519_2.fastq.gz  
SRR18335439_1.fastq.gz  SRR22388518_2.fastq.gz
```

Finalmente, regresemos al directorio de proyectos:

```
cd $HOME/proyectos
```

²Wikipedia - FASTQ Format: https://en.wikipedia.org/wiki/FASTQ_format

3. Control de calidad y limpieza de datos

Una vez descargados los datos, el siguiente paso consiste en hacer control de calidad y limpieza de los datos. En este paso primero se revisa la calidad de los datos, y luego se editan o eliminan las lecturas (*reads*) que no tienen la calidad necesaria para conseguir un buen ensamblaje.

! Importante

Es importante comprender la calidad de los datos de secuenciación. Puede aprender sobre el valor Phred y cómo se codifica en el formato FASTQ en los artículos de Wikipedia "*Nivel de calidad Phred*" y "*FASTQ format*", respectivamente.

3.1. Software requerido

Software	Versión	Descripción	Ref.
FastQC	0.12.1	Herramienta para el control de calidad en datos de secuenciación de alto rendimiento	Sitio Web
MultiQC	1.14	Herramienta para unificar y resumir múltiples reportes generados por FastQC y muchos otros programas de bioinformática.	Ewels et al. (2016); sitio Web
fastp	0.23.2	Herramienta para el pre-procesamiento rápido de archivos Fastq.	Chen et al. (2018); repositorio GitHub

3.2. Determinación de calidad

Para revisar la calidad de los datos descargados, usaremos `fastqc`:

```
# Entrar al directorio de datos en la carpeta del proyecto ensamblaje
cd $HOME/proyectos/ensamblaje/datos
# Analizar todos los archivos descargados
fastqc -o qc/fastqc *.fastq.gz
```

A continuación verá mensajes de progreso de `fastqc`:

```
Started analysis of SRR18335438_1.fastq.gz
Approx 5% complete for SRR18335438_1.fastq.gz
Approx 10% complete for SRR18335438_1.fastq.gz
Approx 15% complete for SRR18335438_1.fastq.gz
Approx 20% complete for SRR18335438_1.fastq.gz
Approx 25% complete for SRR18335438_1.fastq.gz
Approx 30% complete for SRR18335438_1.fastq.gz
Approx 35% complete for SRR18335438_1.fastq.gz
Approx 40% complete for SRR18335438_1.fastq.gz
Approx 45% complete for SRR18335438_1.fastq.gz
Approx 50% complete for SRR18335438_1.fastq.gz
Approx 55% complete for SRR18335438_1.fastq.gz
Approx 60% complete for SRR18335438_1.fastq.gz
Approx 65% complete for SRR18335438_1.fastq.gz
Approx 70% complete for SRR18335438_1.fastq.gz
Approx 75% complete for SRR18335438_1.fastq.gz
Approx 80% complete for SRR18335438_1.fastq.gz
Approx 85% complete for SRR18335438_1.fastq.gz
Approx 90% complete for SRR18335438_1.fastq.gz
Approx 95% complete for SRR18335438_1.fastq.gz
Analysis complete for SRR18335438_1.fastq.gz
...
```

Al final, por cada archivo `.fastq` analizado, `fastqc` genera un archivo `.html` con el reporte de calidad y un archivo comprimido (`.zip`):

```
ls qc/fastqc
```

3. Control de calidad y limpieza de datos

```
SRR18335438_1_fastqc.html SRR18335439_2_fastqc.html
↳ SRR22388519_1_fastqc.html
SRR18335438_1_fastqc.zip SRR18335439_2_fastqc.zip
↳ SRR22388519_1_fastqc.zip
SRR18335438_2_fastqc.html SRR22388518_1_fastqc.html
↳ SRR22388519_2_fastqc.html
SRR18335438_2_fastqc.zip SRR22388518_1_fastqc.zip
↳ SRR22388519_2_fastqc.zip
SRR18335439_1_fastqc.html SRR22388518_2_fastqc.html
SRR18335439_1_fastqc.zip SRR22388518_2_fastqc.zip
```



Nota

El archivo comprimido `.zip` contiene una copia del reporte `.html`, gráficos, y archivos de estadísticas.


3.2.1. Reporte de calidad FastQC

Analicemos uno de los reportes de calidad de FastQC. Abra el archivo [SRR18335438_1_fastqc.html](#). El reporte contiene diferentes secciones, llamadas módulos, que corresponden a diferentes evaluaciones de calidad que la herramienta hace sobre los datos (Figura 3.1). Puede consultar información detallada (en inglés) a cerca de FastQC y sus módulos en el Apéndice A. Por cada módulo, el reporte presenta un ícono con tres posibles estados de acuerdo al resultado de la evaluación (Tabla 3.2).

Tabla 3.2.: Posibles resultados de la evaluación de calidad realizada por la herramienta FastQC

Símbolo	Descripción
	Pasó. La muestra pasó la evaluación de calidad en esta categoría (módulo). No hay problemas de calidad.
	Advertencia. La muestra tiene un posible problema calidad. El usuario debería revisar la muestra y tomar las medidas necesarias para evitar el error de ser posible. Por la naturaleza de la secuenciación de ADN, es normal que se presenten advertencias en algunos módulos en las muestras sin que signifique que haya un problema real o grave.

3. Control de calidad y limpieza de datos

Símbolo	Descripción
	Falló. La muestra falló la evaluación de calidad en este módulo. El usuario debería limpiar los datos para corregir el error.

De acuerdo con los estados, vemos que las muestras fallaron las evaluaciones de calidad de los módulos “*Per base sequence quality*” y “*Sequence Duplication Levels*”, y el módulo “*Sequence Length Distribution*” tiene una advertencia (Figura 3.1).

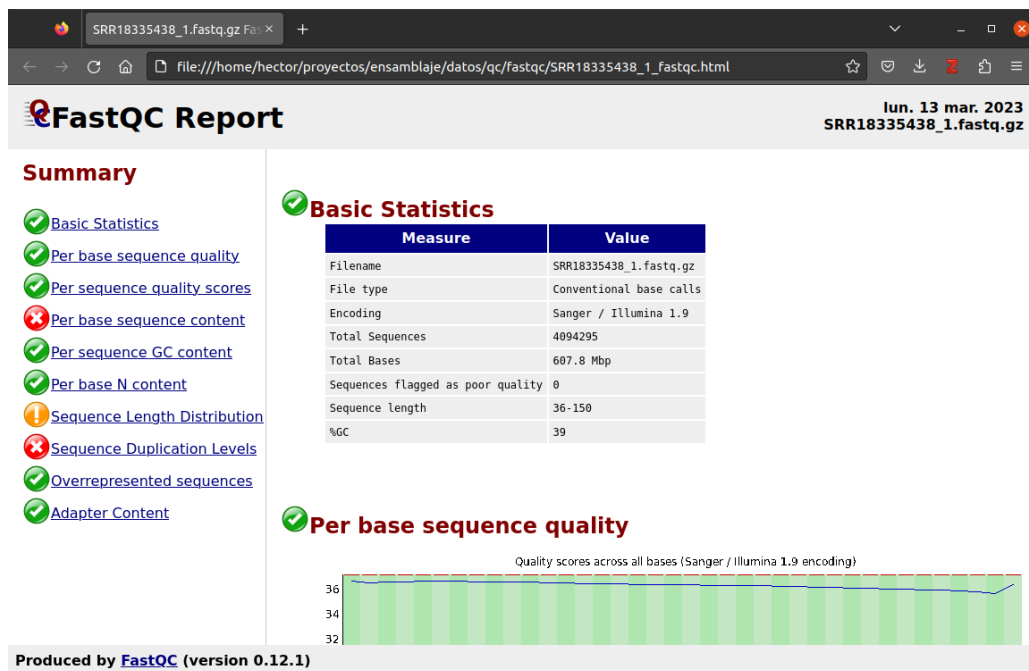


Figura 3.1.: Captura de pantalla de un reporte de calidad generado con la aplicación FastQC.

Tal vez el módulo más importante a revisar es “*Per base sequence quality*” (Sección A.3.5), el cual muestra el valor de calidad de por cada posición a lo largo de las secuencias del archivo procesado. Típicamente en archivos producidos con tecnología Illumina, la calidad es baja en las primeras siete a 10 bases de las secuencias, incrementando hasta alcanzar su máximo hacia la parte media de la secuencia. Luego la calidad empieza a decrecer hasta encontrar sus mínimos valores al final de la secuencia. Comúnmente la calidad mínima para lograr ensamblajes decentes es $Q \geq 20$ (eje Y en el gráfico). Así, este gráfico es crucial para que el usuario decida el número de bases (eje X) que se recortarán en los extremos de

3. Control de calidad y limpieza de datos

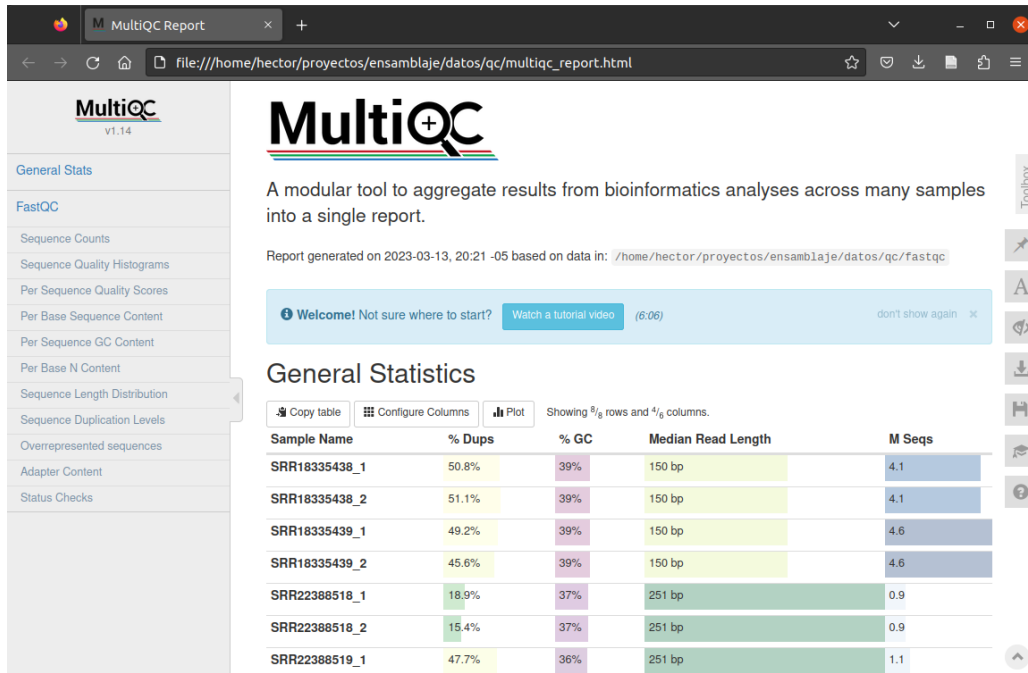


Figura 3.2.: Captura de pantalla del reporte de calidad unificado generado con MultiQC.

Echemos un vistazo ahora al gráfico de calidad de secuenciación en la sección “*Sequence Quality Histograms*” (Figura 3.3). Además de la diferencia de longitud de secuencias, se puede notar que:

- La calidad al inicio de las secuencias en todos los archivos es buena ($Q \geq 30$)
- La calidad al final de las secuencias es aceptable ($Q \geq 20$) en todos los archivos, con excepción de las secuencias *reverse* (_2) de las muestras SRR22388518 y SRR22388519.

De estas observaciones se puede concluir que la limpieza de los datos consistirá en recortar las últimas bases (aproximadamente 20) de los archivos SRR22388518_2.fastq.gz y SRR22388519_2.fastq.gz pues son las que presentan baja calidad. Además, como buena práctica, se eliminarán aquellas secuencias cuya calidad promedio sea $Q < 20$.

3.3. Limpieza

Usaremos la aplicación fastp para recortar y filtrar las secuencias por calidad. Empecemos procesando la primera muestra:

3. Control de calidad y limpieza de datos

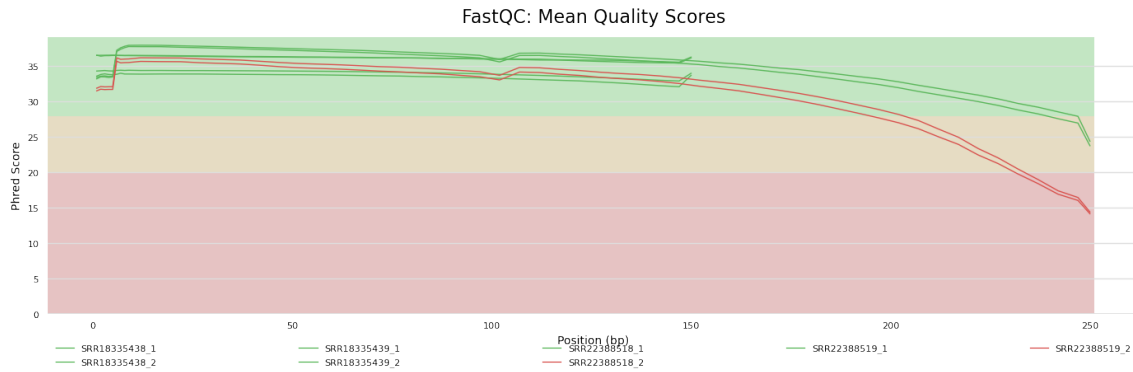


Figura 3.3.: Gráfico de calidad de secuenciación por base

```
# Entrar al directorio raíz del proyecto (en mi caso,
↳ /home/hector/ensamblaje)
cd ..

# Comando para ejecutar la limpieza de la primera muestra
fastp
  --verbose \
  --thread 4 \
  --detect_adapter_for_pe \
  --cut_tail \
  --cut_mean_quality 20 \
  --average_qual 20 \
  --report_title 'Reporte fastp: SRR22388518' \
  --in1 datos/SRR22388518_1.fastq.gz \
  --in2 datos/SRR22388518_2.fastq.gz \
  --out1 resultados/00_datos_limpios/SRR22388518_1.clean.fastq.gz \
  --out2 resultados/00_datos_limpios/SRR22388518_2.clean.fastq.gz \
  --unpaired1
  ↳ resultados/00_datos_limpios/SRR22388518_1.unpaired.fastq.gz \
  --unpaired2
  ↳ resultados/00_datos_limpios/SRR22388518_2.unpaired.fastq.gz \
  --html resultados/00_datos_limpios/qc/fastp/SRR22388518.report.fastp.
  ↳ html
  ↳ \
```

3. Control de calidad y limpieza de datos

```
--json resultados/00_datos_limpios/qc/fastp/SRR22388518.report.fastp. |  
↵ json
```

Antes de revisar el resultado, veamos las opciones usadas con el comando y su finalidad:

- **verbose**: provee información adicional durante el procesamiento de los archivos
- **thread**: usa cuatro hilos de ejecución de manera paralela para acelerar el procesamiento de los archivos
- **detect_adapter_for_pe**: detecta las secuencias de adaptadores de secuenciación y las elimina
- **cut_tail**: corta la “cola” (*tail*) de las secuencias, la cual se dijo tiene las bases de más baja calidad
- **cut_mean_quality**: la calidad media mínima aceptada para corte de secuencias. *fastp* usa una ventana de 4 bases desde el extremo de la secuencia para revisar calidad; si el promedio de estas cuatro bases está por debajo de 20, elimina este extremo.
- **average_qual**: la calidad media mínima aceptada para mantener una secuencia. Si al calcular la calidad media de toda la secuencia está por debajo de 20, la secuencia es descartada.
- **report_title**: el título del reporte
- **in1**: el archivo de secuencias *forward* a ser procesado
- **in2**: el archivo de secuencias *reverse* a ser procesado
- **out1**: el archivo de salida con las secuencias *forward* que pasaron el control de calidad
- **out2**: el archivo de salida con las secuencias *reverse* que pasaron el control de calidad
- **unpaired1**: el archivo de salida con las secuencias *forward* no pareadas que pasaron el control de calidad (no tienen una secuencia par *reverse*)
- **unpaired2**: el archivo de salida con las secuencias *reverse* no pareadas que pasaron el control de calidad (no tienen una secuencia par *forward*)
- **html**: el archivo de reporte en formato html
- **json**: el archivo de reporte en formato json (estadísticas)

Tip

Para conocer todas las opciones disponibles de *fastp* consulte la ayuda ejecutando `fastp --help`.

En el anterior comando, en lugar de cortar un número determinado de bases al final de las secuencias, se usa la estrategia del programa *fastp*. Por cada secuencia del archivo, *fastp* revisa la calidad media de una pequeña porción de 4 bases, llamado ventana, empezando desde el extremo final. Si la calidad media está por debajo del mínimo definido (20 en este caso), *fastp* recorta esta ventana de la secuencia. Este proceso se repita hasta que no

3. Control de calidad y limpieza de datos

se encuentre una ventana que deba recortarse. Así, solo las secuencias con baja calidad al final son recortadas. Este mismo principio se puede aplicar para el inicio de las secuencias, cuando sea el caso.

Veamos ahora el resultado de fastp. Al ejecutar el anterior comando veremos una salida parecida a esto:

```
Detecting adapter sequence for read1...
>Illumina TruSeq Adapter Read 1
AGATCGGAAGAGCACACGTCTGAACTCCAGTCA

Detecting adapter sequence for read2...
No adapter detected for read2

[19:24:32] start to load data of read1
[19:24:32] start to load data of read2
[19:24:36] Read1: loading completed with 871 packs
[19:24:36] Read2: loading completed with 871 packs
[19:24:39] thread 4 data processing completed
[19:24:39] thread 4 finished
[19:24:39] thread 3 data processing completed
[19:24:39] thread 3 finished
[19:24:39] thread 2 data processing completed
[19:24:39] thread 2 finished
[19:24:39] thread 1 data processing completed
[19:24:39] thread 1 finished
[19:24:39] resultados/00_datos_limpios/SRR22388518_1.unpaired.fastq.gz
↳ writer finished
[19:24:39] resultados/00_datos_limpios/SRR22388518_2.unpaired.fastq.gz
↳ writer finished
[19:24:40] resultados/00_datos_limpios/SRR22388518_2.clean.fastq.gz
↳ writer finished
[19:24:40] resultados/00_datos_limpios/SRR22388518_1.clean.fastq.gz
↳ writer finished
[19:24:40] start to generate reports

Read1 before filtering:
total reads: 870940
total bases: 218350507
Q20 bases: 198719240(91.0093%)
Q30 bases: 184185641(84.3532%)
```

3. Control de calidad y limpieza de datos

Read2 before filtering:
total reads: 870940
total bases: 218399177
Q20 bases: 175386628(80.3055%)
Q30 bases: 149084629(68.2624%)

Read1 after filtering:
total reads: 863093
total bases: 209876970
Q20 bases: 193175278(92.0422%)
Q30 bases: 180122777(85.823%)

Read2 after filtering:
total reads: 863093
total bases: 201834951
Q20 bases: 170111238(84.2823%)
Q30 bases: 146895463(72.78%)

Filtering result:
reads passed filter: 1726186
reads failed due to low quality: 12556
reads failed due to too many N: 488
reads failed due to too short: 2650
reads with adapter trimmed: 150684
bases trimmed due to adapters: 7399476

Duplication rate: 0.0137782%

Insert size peak (evaluated by paired-end reads): 251

JSON report: resultados/00_datos_limpios/qc/fastp/report.fastp.json
HTML report: resultados/00_datos_limpios/qc/fastp/report.fastp.html

3. Control de calidad y limpieza de datos

```
fastp --verbose --thread 4 --detect_adapter_for_pe --cut_tail
↳ --cut_mean_quality 20 --average_qual 20 --report_title Reporte
↳ fastp: SRR22388518 --in1 datos/SRR22388518_1.fastq.gz --in2
↳ datos/SRR22388518_2.fastq.gz --out1
↳ resultados/00_datos_limpios/SRR22388518_1.clean.fastq.gz --out2
↳ resultados/00_datos_limpios/SRR22388518_2.clean.fastq.gz
↳ --unpaired1
↳ resultados/00_datos_limpios/SRR22388518_1.unpaired.fastq.gz
↳ --unpaired2
↳ resultados/00_datos_limpios/SRR22388518_2.unpaired.fastq.gz --html
↳ resultados/00_datos_limpios/qc/fastp/SRR22388518.report.fastp.html
↳ --json
↳ resultados/00_datos_limpios/qc/fastp/SRR22388518.report.fastp.json
fastp v0.23.2, time used: 22 seconds
```

Podemos encontrar información importante en los mensajes de la salida, como por ejemplo el número de secuencias y calidad antes y después de aplicar la limpieza o filtrado (secciones `Read1 before filtering:`, `Read2 before filtering:`, `Read1 after filtering:` y `Read2 after filtering:`), y el resumen del número de reads que pasaron y no pasaron el filtro (sección `Filtering result`).

Junto con los archivos de secuencias filtradas, fastp genera un archivo de reporte `.html` y un archivo de estadísticas.

Ahora se deben procesar el resto de los archivos:

```
fastp
  --verbose \
  --thread 4 \
  --detect_adapter_for_pe \
  --cut_tail \
  --cut_mean_quality 20 \
  --average_qual 20 \
  --report_title 'Reporte fastp: SRR22388519' \
  --in1 datos/SRR22388519_1.fastq.gz \
  --in2 datos/SRR22388519_2.fastq.gz \
  --out1 resultados/00_datos_limpios/SRR22388519_1.clean.fastq.gz \
  --out2 resultados/00_datos_limpios/SRR22388519_2.clean.fastq.gz \
  --unpaired1
  ↳ resultados/00_datos_limpios/SRR22388519_1.unpaired.fastq.gz \
```


3. Control de calidad y limpieza de datos

```
--unpaired2
↳ resultados/00_datos_limpios/SRR22388519_2.unpaired.fastq.gz \
--html resultados/00_datos_limpios/qc/fastp/SRR22388519.report.fastp.
↳ html
↳ \
--json resultados/00_datos_limpios/qc/fastp/SRR22388519.report.fastp.
↳ json
```

fastp

```
--verbose \
--thread 4 \
--detect_adapter_for_pe \
--cut_tail \
--cut_mean_quality 20 \
--average_qual 20 \
--report_title 'Reporte fastp: SRR18335438' \
--in1 datos/SRR18335438_1.fastq.gz \
--in2 datos/SRR18335438_2.fastq.gz \
--out1 resultados/00_datos_limpios/SRR18335438_1.clean.fastq.gz \
--out2 resultados/00_datos_limpios/SRR18335438_2.clean.fastq.gz \
--unpaired1
↳ resultados/00_datos_limpios/SRR18335438_1.unpaired.fastq.gz \
--unpaired2
↳ resultados/00_datos_limpios/SRR18335438_2.unpaired.fastq.gz \
--html resultados/00_datos_limpios/qc/fastp/SRR18335438.report.fastp.
↳ html
↳ \
--json resultados/00_datos_limpios/qc/fastp/SRR18335438.report.fastp.
↳ json
```

fastp

```
--verbose \
--thread 4 \
--detect_adapter_for_pe \
--cut_tail \
--cut_mean_quality 20 \
--average_qual 20 \
--report_title 'Reporte fastp: SRR18335439' \
--in1 datos/SRR18335439_1.fastq.gz \
```

3. Control de calidad y limpieza de datos

```
--in2 datos/SRR18335439_2.fastq.gz \  
--out1 resultados/00_datos_limpios/SRR18335439_1.clean.fastq.gz \  
--out2 resultados/00_datos_limpios/SRR18335439_2.clean.fastq.gz \  
--unpaired1  
  ↪ resultados/00_datos_limpios/SRR18335439_1.unpaired.fastq.gz \  
--unpaired2  
  ↪ resultados/00_datos_limpios/SRR18335439_2.unpaired.fastq.gz \  
--html resultados/00_datos_limpios/qc/fastp/SRR18335439.report.fastp.\  
  ↪ html  
  ↪ \  
--json resultados/00_datos_limpios/qc/fastp/SRR18335439.report.fastp.\  
  ↪ json
```

3.4. Verificación de calidad después de la limpieza

Finalmente podemos usar FastQC y MultiQC para revisar la calidad de las muestras después de la limpieza:

```
# Generar reportes con FastQC  
fastqc \  
  -o resultados/00_datos_limpios/qc/fastqc \  
  resultados/00_datos_limpios/*.clean.fastq.gz  
  
# Generar reporte unificado con MultiQC  
multiqc \  
  --filename multiqc_report.html \  
  --outdir resultados/00_datos_limpios/qc \  
  resultados/00_datos_limpios/qc/fastp \  
  resultados/00_datos_limpios/qc/fastqc
```

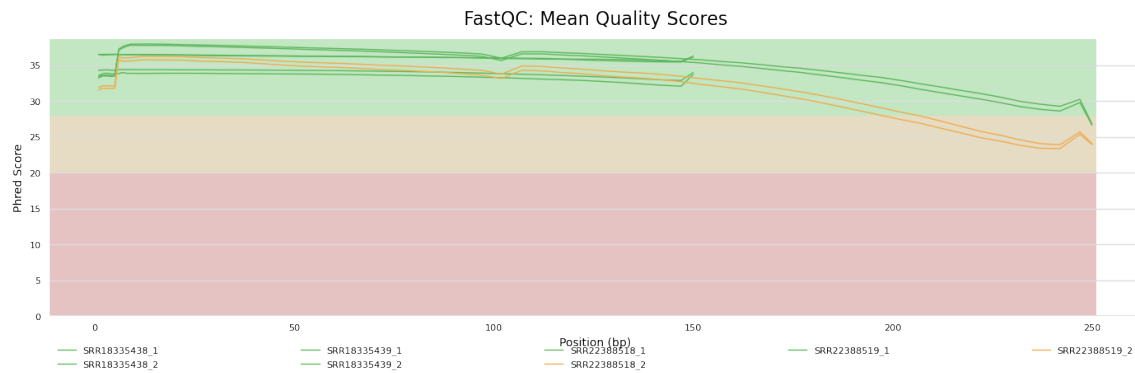
Note que en el anterior comando de MultiQC se pasó no solo el directorio de reportes FastQC, sino también el directorio con los reportes de fastp. La aplicación se encarga de generar un solo documento `.html` con dos secciones, una para fastp y otra para FastQC.

3. Control de calidad y limpieza de datos

Tip

Explore el reporte [resultados/00_datos_limpios/qc/multiqc_report.html](#) para ver las diferentes estadísticas de calidad después de haber aplicado los filtros a las muestras.

Echemos un vistazo al gráfico de calidad de secuenciación por base generado sobre los datos limpios (Figura 3.3). Podemos ver que las dos muestras que te tenían baja calidad hacia el final de las secuencias, ahora aparecen en color naranja, con una calidad $Q > 20$.



Finalmente las muestras están listas para ser ensambladas.

4. Ensamblaje

Ahora que los datos están depurados, podemos ensamblar las muestras en sus correspondientes genomas

4.1. Software requerido

Software	Versión	Descripción	Ref.
SPAdes	3.15.5	Conjunto de herramientas para ensamblaje <i>de novo</i> de genomas con múltiples flujos de trabajo de ensamblaje especializado.	Prjibelski et al. (2020); Sitio Web ; repositorio GitHub

4.2. Ensamblaje de genomas

Usaremos el ensamblador *de novo*¹ SPAdes (Prjibelski et al. 2020), el cual fue desarrollado inicialmente para ensamblar secuencias cortas Illumina obtenidas de aislamientos bacterianos y secuenciación de células individuales (*single-cell sequencing*), y que con el tiempo ha sido extendido para usar secuencias de otras tecnologías y ensamblar metagenomas, transcriptomas, y plásmidos.

Para ensamblar la muestra ejecute la siguiente orden:

```
# Entrar en el directorio raíz del proyecto
cd ensamblaje
# Ejecutar el ensamblaje
spades.py \
  --isolate \
  -l resultados/00_datos_limpios/SRR22388518_1.clean.fastq.gz \
```

¹Wikipedia - *De novo assemblers*: https://en.wikipedia.org/wiki/De_novo_sequence_assemblers

4. Ensamblaje

```
-2 resultados/00_datos_limpios/SRR22388518_2.clean.fastq.gz \  
-o resultados/01_ensamblaje/SRR22388518 \  
-t 4
```

Veamos los parámetros usados en el comando anterior:

- `isolate`: indica a SPAdes que el *pipeline* a ser ejecutado es el de ensamblaje de aislamientos bacterianos.
- `1`: archivo con los *reads forward*.
- `2`: archivo con los *reads reverse*.
- `o`: directorio donde se almacenarán los archivos de salida.
- `t`: hilos (*threads*) de procesos en paralelo para incrementar la velocidad del ensamblaje.

El tiempo de ensamblaje depende del tamaño de la muestra a ensamblar, es decir de número de *reads* en los archivos `.fastq.gz` de entrada, y del número de *cores* o procesadores del computador y los hilos asignados al comando (opción `-t`). En el caso anterior el proceso tomó 35 minutos para ensamblar un total de 1726.186 de *reads* (863093×2) usando 4 hilos en una máquina con ocho *cores*.

Ahora veamos la salida del comando:

```
Command line: /home/hector/mambaforge/envs/ensam/bin/spades.py  
↳ --isolate -1 /home/hector/proyectos/ensamblaje/resultados/00_dat  
↳ os_limpios/SRR22388518_1.clean.fastq.gz -2  
↳ /home/hector/proyectos/ensamblaje/resultados/00_datos_limpios/SRR2  
↳ 2388518_2.clean.fastq.gz -o  
↳ /home/hector/proyectos/ensamblaje/resultados/01_ensamblaje/SRR2238  
↳ 8518 -t  
↳ 4
```

System information:

```
SPAdes version: 3.15.5  
Python version: 3.10.6  
OS: Linux-5.15.0-67-generic-x86_64-with-glibc2.31
```

```
Output dir: /home/hector/proyectos/ensamblaje/resultados/01_ensamblaje  
↳ /SRR22388518
```

```
Mode: ONLY assembling (without read error correction)
```


4. Ensamblaje

```
* Paths in the assembly graph corresponding to the scaffolds are in
↳ /home/hector/proyectos/ensamblaje/resultados/01_ensamblaje/SRR223_
↳ 88518/scaffolds.paths
* Assembly graph is in /home/hector/proyectos/ensamblaje/resultados/01_
↳ _ensamblaje/SRR22388518/assembly_graph.fastg
* Assembly graph in GFA format is in
↳ /home/hector/proyectos/ensamblaje/resultados/01_ensamblaje/SRR223_
↳ 88518/assembly_graph_with_scaffolds.gfa
```

```
=====  
SPAdes pipeline finished.
```

```
SPAdes log can be found here: /home/hector/proyectos/ensamblaje/resulta_
↳ dos/01_ensamblaje/SRR22388518/spades.log
```

Thank you for using SPAdes!

En la salida anterior, se puede observar al principio el comando ejecutado, así como información del sistema y la ubicación de los datos de entrada y salida. `spades.py` indica el inicio del ensamblaje mediante el encabezado `=====
SPAdes pipeline started.` y presenta la ubicación del archivo `log` (`Log can be found here: ...`). Este archivo `log` es muy importante pues contiene toda la información del proceso del ensamblaje, incluido cualquier error que se haya presentado, lo que permite tomar acciones para corregirlo. `spades.py` también detalla el proceso de ensamblaje a través de una amplia serie de mensajes ubicados en lugar de los tres puntos `...` en la salida anterior. Cuando el proceso de ensamblaje es exitoso, `spades.py` presenta la sección `=====
Terminate finished.` que contiene la ubicación de los archivos de salida. Finalmente, bajo la sección `=====
SPAdes pipeline finished.`, el programa presenta de nuevo la ubicación del archivo `log`.

Examinemos ahora el directorio de salida de SPAdes:

```
ls resultados/01_ensamblaje/SRR22388518

assembly_graph_after_simplification.gfa
assembly_graph.fastg
assembly_graph_with_scaffolds.gfa
before_rr.fasta
contigs.fasta
contigs.paths
dataset.info
input_dataset.yaml
```

4. Ensamblaje

```
K127
K21
K33
K55
K77
K99
misc
params.txt
pipeline_state
run_spades.sh
run_spades.yaml
scaffolds.fasta
scaffolds.paths
spades.log
tmp
```

Estos son los archivos de salida cuando el ensamblaje es exitoso:

- `contigs.fasta`: las secuencias de los *contigs*² (en español, llamado menos frecuentemente como *cóntigo*) en formato FASTA³
- `scaffolds.fasta`: las secuencias de *scaffolds*⁴ en formato FASTA
- `assembly_graph.gfa`: grafo de ensamblaje y los caminos de los *scaffolds* en formato GFA 1.0⁵
- `assembly_graph.fastg`: grafo del ensamblaje en formato FASTG
- `contigs.paths`: caminos de los *contigs* en el grafo de ensamblaje
- `scaffolds.paths`: caminos de los *scaffolds* en el grafo de ensamblaje
- `spades.log`: archivo con todos los mensajes del proceso

Dado que SPAdes es un ensamblador de genomas basado en grafos de De Bruijn⁶ (Compeau, Pevzner, y Tesler 2011), algunos de los archivos de salida contienen información de los grafos usados durante el ensamblaje (con extensión `.gfa` y `.fastg`). Si desea visualizar los archivos de grafos puede usar la aplicación Bandage⁷ (Wick et al. 2015).

Los archivos de salida más importantes son aquellos que contienen las secuencias del ensamblaje en sí, es decir los *contigs* y *scaffolds* en formato FASTA (`.fasta`). Los *scaffolds* son

²Wikipedia - C3ntigo: <https://es.wikipedia.org/wiki/C%C3%B3ntigo>

³Wikipedia - Formato FASTA: https://es.wikipedia.org/wiki/Formato_FASTA

⁴Wikipedia - Scaffolding (bioinformatics): [https://en.wikipedia.org/wiki/Scaffolding_\(bioinformatics\)](https://en.wikipedia.org/wiki/Scaffolding_(bioinformatics))

⁵The GFA Format Specification: <https://gfa-spec.github.io/GFA-spec/GFA1.html>

⁶Wikipedia - Grafo de De Bruijn: https://es.wikipedia.org/wiki/Grafo_de_De_Bruijn

⁷Bandage: <http://rrwick.github.io/Bandage/>

4. Ensamblaje

las secuencias más largas de ensamblaje, las cuales son creadas a partir de las secuencias de los *contigs*. Este es el archivo que se usará como resultado final del ensamblaje y por lo tanto anotado.

Veamos las primeras 10 líneas del archivo `scaffolds.fasta`:

```
head resultados/01_ensamblaje/SRR22388518/scaffolds.fasta

>NODE_1_length_407393_cov_61.379170
GCTCTGTTGAATGATTTATCCAAAAATCGTTTAGCAACTAGCAAGGTTGAACTAGGAATA
CAACAAAAATCCTAAATTAAGAGAATTTTATAGCTTTTTATAGGATTTTATAGGTTTGTA
GTAACGGTATTTTGGGCAAGAAATCAACCTGACCACTTACAACCTTTGCCGCATCAACAT
GTTTTTGCATGACATCAATTACTCTAAATTCACATTGCGCTAGGGGACACGCTTTTAGA
TCCAAAACATGAAGACGATGAGCCTTTTGATGCGATCGTTTCCAACCTCCTTATTCCAC
TAAATGGGTGGGCGATAACAACCTATTTAATCAACGACGAGCGCTTTAGCCCGGCCGG
TGTGCTAGCGCCCAAAAACGCCGCGATCTCGCTTTCACCATGCACATGCTTTCTTATTT
ATCCAATAGCGGCACGGCTGCGATCGTGAATTCCCCGGGTGCTTTATAGGGGGAATGC
TGAAGCAAAAATCAGAGAATATTTAGTCAAAGAAAATTTATTGACTGCGTGATCGCTTT
```

En el identificador (nombre) del primer *scaffold* (`NODE_1_length_407393_cov_61.379170`) se puede observar alguna información de la secuencia: `1` es el número del *contig/scaffold*, `407393` es la longitud de la secuencia en pares de bases (bp), y `61.379170` es el número de *k*-mers que cubren el *scaffold*.

Con un simple comando podemos averiguar el número total de *scaffolds* en el ensamblaje:

```
grep -c '>' resultados/01_ensamblaje/SRR22388518/scaffolds.fasta

336
```

Ahora terminaremos de procesar el conjunto de datos ensamblando el resto de las muestras:

```
spades.py \
  --isolate \
  -1 resultados/00_datos_limpios/SRR22388519_1.clean.fastq.gz \
  -2 resultados/00_datos_limpios/SRR22388519_2.clean.fastq.gz \
  -o resultados/01_ensamblaje/SRR22388519 \
  -t 4
```

4. Ensamblaje

```
spades.py \  
  --isolate \  
  -1 resultados/00_datos_limpios/SRR18335437_1.clean.fastq.gz \  
  -2 resultados/00_datos_limpios/SRR18335437_2.clean.fastq.gz \  
  -o resultados/01_ensamblaje/SRR18335437 \  
  -t 4
```

```
spades.py \  
  --isolate \  
  -1 resultados/00_datos_limpios/SRR18335438_1.clean.fastq.gz \  
  -2 resultados/00_datos_limpios/SRR18335438_2.clean.fastq.gz \  
  -o resultados/01_ensamblaje/SRR18335438 \  
  -t 4
```

Para que cada ensamblaje (`scaffolds.fasta`) tenga el nombre de la correspondiente muestra, crearemos enlaces simbólicos en el directorio `resultados/01_ensamblaje`. Por ejemplo, dentro del directorio `resultados/01_ensamblaje` el archivo `SRR22388518.fasta` apuntará al archivo `SRR22388519/scaffolds.fasta`. Esto se hace para que los pasos siguientes del manual (control de calidad y anotación) se ejecuten con mayor comodidad. Para crear los enlaces ejecute la siguientes órdenes:

```
ln -sr resultados/01_ensamblaje/SRR22388518/scaffolds.fasta \  
  resultados/01_ensamblaje/SRR22388518.fasta  
  
ln -sr resultados/01_ensamblaje/SRR22388519/scaffolds.fasta \  
  resultados/01_ensamblaje/SRR22388519.fasta  
  
ln -sr resultados/01_ensamblaje/SRR18335437/scaffolds.fasta \  
  resultados/01_ensamblaje/SRR18335437.fasta  
  
ln -sr resultados/01_ensamblaje/SRR18335438/scaffolds.fasta \  
  resultados/01_ensamblaje/SRR18335438.fasta
```

Verifiquemos los enlaces creados:

```
ls -l resultados/01_ensamblaje  
  
total 16K  
drwxrwxr-x  9 hector hector 4,0K mar 28 13:32 SRR18335438
```

4. Ensamblaje

```
lrwxrwxrwx 1 hector hector 27 mar 28 10:46 SRR18335438.fasta ->
↳ SRR18335438/scaffolds.fasta
drwxrwxr-x 9 hector hector 4,0K mar 28 14:13 SRR18335439
lrwxrwxrwx 1 hector hector 27 mar 28 10:46 SRR18335439.fasta ->
↳ SRR18335439/scaffolds.fasta
drwxrwxr-x 11 hector hector 4,0K mar 28 15:15 SRR22388518
lrwxrwxrwx 1 hector hector 27 mar 28 10:46 SRR22388518.fasta ->
↳ SRR22388518/scaffolds.fasta
drwxrwxr-x 11 hector hector 4,0K mar 28 13:08 SRR22388519
lrwxrwxrwx 1 hector hector 27 mar 28 10:46 SRR22388519.fasta ->
↳ SRR22388519/scaffolds.fasta
```

Finalmente, con todos los ensamblajes listos, procederemos a analizar su calidad.

5. Control de calidad de ensamblajes

En este paso se calcularán sobre cada ensamblaje, varias métricas que pueden ser usadas como indicadores de su calidad.

5.1. Software requerido

Software	Versión	Descripción	Ref.
Quast	5.2.0	Herramienta para evaluar la calidad de ensamblajes de genomas (pequeños) y metagenomas, mediante el cálculo de diferentes métricas.	Gurevich et al. (2013); Sitio Web ; repositorio GitHub

5.2. Determinación de calidad

EL cálculo de las métricas se realiza usando el software Quast (del inglés *Quality Assessment Tool*). Aunque esta herramienta funciona sin un genoma de referencia, la evaluación de calidad será mejor si se provee uno. Este genoma podrá ser el del organismo más cercano o, como en nuestro caso, el del propio organismo (*H. pylori*¹). Como tenemos acceso a un genoma de referencia, lo usaremos.

¹NCBI Datasets Genome page of *Helicobacter pylori*: https://www.ncbi.nlm.nih.gov/labs/data-hub/genome/GCF_017821535.1/

5. Control de calidad de ensamblajes

5.2.1. Descarga del genoma de referencia

El número de acceso (*accession number*) del genoma de referencia de *H. pylori*, es GCF_017821535.1. Este número es necesario para descargar el archivo del genoma con la herramienta `datasets`:

```
# Entrar al directorio datos del proyecto
cd $HOME/proyectos/ensamblaje/datos

# Descargar el paquete de datos con el genoma
datasets download genome accession GCF_017821535.1 --filename
↳ GCF_017821535.1_dataset.zip

Collecting 1 records
↳ [=====] 100% 1/1
Downloading: GCF_017821535.1_dataset.zip 473kB done
```

Descargado el paquete de datos con el genoma, debemos descomprimirlo:

```
unzip GCF_017821535.1_dataset.zip

Archive: GCF_017821535.1_dataset.zip
  inflating: README.md
  inflating: ncbi_dataset/data/assembly_data_report.jsonl
  inflating: ncbi_dataset/data/GCF_017821535.1/GCF_017821535.1_ASM17821_
↳ 53v1_genomic.fna
  inflating: ncbi_dataset/data/dataset_catalog.json
```

Al descomprimir el paquete se generaron dos elementos, el archivo `README.md` y el directorio `ncbi_dataset`. El archivo que necesitamos es `ncbi_dataset/data/GCF_017821535.1/GCF_017821535.1_ASM1782153v1_genomic.fna`, así que lo moveremos y renombraremos a la ubicación apropiada:

```
mv ncbi_dataset/data/GCF_017821535.1/GCF_017821535.1_ASM1782153v1_genomic.fna
↳ mic.fna
↳ \
GCF_017821535.1.fasta
```

Ahora podemos eliminar los archivos que no usaremos:

5. Control de calidad de ensamblajes

```
rm -fr README.md ncbi_dataset GCF_017821535.1_dataset.zip
```

5.2.2. Ejecución de Quast

```
cd $HOME/proyectos/ensamblaje
```

```
quast -o resultados/02_ensamblaje_qc/quast \  
-r datos/GCF_017821535.1.fasta \  
-t 4 \  
resultados/01_ensamblaje/*.fasta
```

```
/home/hector/mambaforge/envs/ensam/bin/quast -o  
↳ resultados/02_ensamblaje_qc/quast -r datos/GCF_017821535.1.fasta -t  
↳ 4 resultados/01_ensamblaje/SRR18335438.fasta  
↳ resultados/01_ensamblaje/SRR18335439.fasta  
↳ resultados/01_ensamblaje/SRR22388518.fasta  
↳ resultados/01_ensamblaje/SRR22388519.fasta
```

```
Version: 5.2.0
```

```
System information:
```

```
OS: Linux-5.15.0-67-generic-x86_64-with-glibc2.31 (linux_64)
```

```
Python version: 3.10.6
```

```
CPUs number: 8
```

```
Started: 2023-03-13 21:09:50
```

```
Logging to /home/hector/proyectos/ensamblaje/resultados/02_ensamblaje_  
↳ qc/quast/quast.log
```

```
NOTICE: Output directory already exists and looks like a QFAST output
```

```
↳ dir. Existing results can be reused (e.g. previously generated  
↳ alignments)!
```

```
...
```

```
RESULTS:
```

5. Control de calidad de ensamblajes

```
Text versions of total report are saved to /home/hector/proyectos/ensamblaje/resultados/02_ensamblaje_qc/quast/report.txt, report.tsv, and report.tex
Text versions of transposed total report are saved to /home/hector/proyectos/ensamblaje/resultados/02_ensamblaje_qc/quast/transposed_report.txt, transposed_report.tsv, and transposed_report.tex
HTML version (interactive tables and plots) is saved to /home/hector/proyectos/ensamblaje/resultados/02_ensamblaje_qc/quast/report.html
PDF version (tables and plots) is saved to /home/hector/proyectos/ensamblaje/resultados/02_ensamblaje_qc/quast/report.pdf
Icarus (contig browser) is saved to /home/hector/proyectos/ensamblaje/resultados/02_ensamblaje_qc/quast/icarus.html
Log is saved to /home/hector/proyectos/ensamblaje/resultados/02_ensamblaje_qc/quast/quast.log
```

```
Finished: 2023-03-13 21:10:00
Elapsed time: 0:00:10.278012
NOTICES: 4; WARNINGS: 0; non-fatal ERRORS: 0
```

Thank you for using QUAST!

Quast genera un reporte en diferentes formatos: texto (`.txt` y `.tsv`), LaTeX (`.tex`), html (`.html`) y pdf (`.pdf`).

5.3. Evaluando la calidad

Demos un vistazo al reporte `.html` ubicado en `ensamblaje/resultados/02_ensamblaje_qc/quast/report.html` (Figura 5.1).

El reporte `.html` presenta una tabla con las estadísticas de los cuatro ensamblajes analizados. Quast además adiciona colores en forma de mapa de calor a la tabla para identificar rápidamente el mejor/peor (azul/rojo) ensamblaje con respecto de una estadística determinada.

Advertencia

La guía del mapa de calor es solo visual y no implica que los ensamblajes estén buenos o malos, pues la calidad de un ensamblaje está dada por una combinación de métricas.

5. Control de calidad de ensamblajes

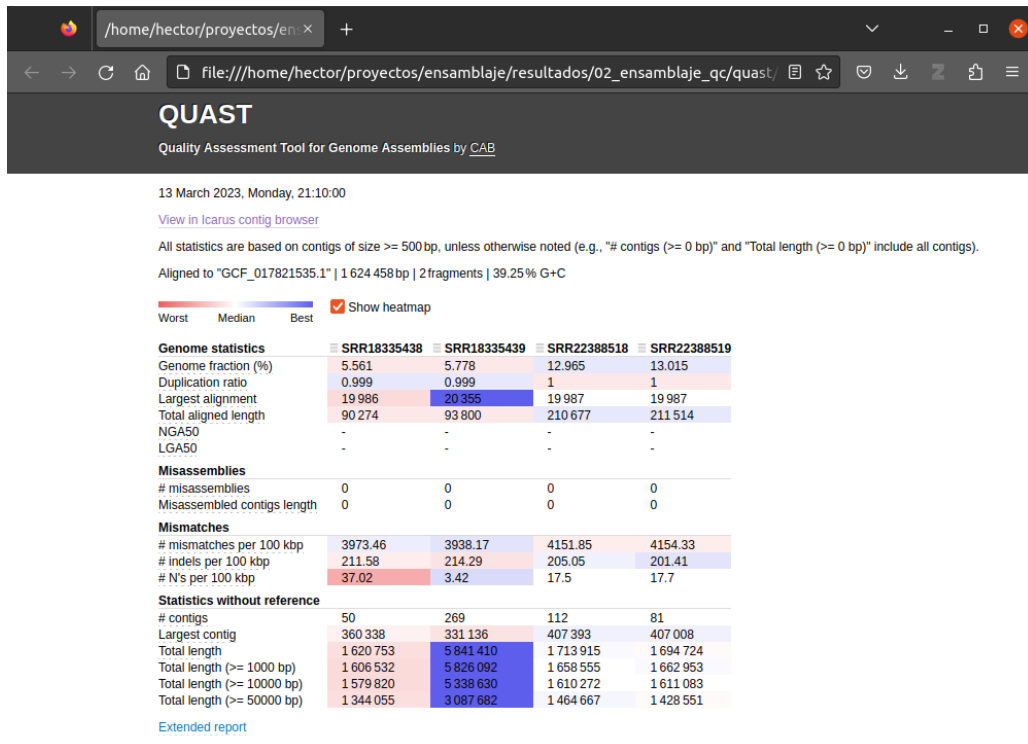


Figura 5.1.: Reporte HTML generado por Quast.

5. Control de calidad de ensamblajes

Tip

Puede ver la definición (en inglés) de cada estadística ubicando el puntero sobre el nombre correspondiente.

Si hace clic en “*Extended report*” verá aún más estadísticas, algunas de ellas bastante importantes. Entre las estadísticas más importantes para tener en cuenta se encuentran:

- **# contigs**: el número total de *contigs* con longitud mayor a 500 bp (*base pairs* o pares de bases). Cuanto menor número de contigs obtenidos, mejor será en ensamblaje.
- **Total length**: la suma de las longitudes de los *contigs* en el ensamblaje. Es importante conocer el tamaño del genoma del organismo que se estudia para evaluar la calidad del ensamblaje. Se requiere que la longitud total obtenida sea cercana a la longitud del genoma del organismo estudiado. En nuestro caso, el genoma de *H. pylori* tiene un tamaño estimado de 1.6 Mbp (*Mega base pairs*).
- **N50**: es la longitud x del *contig* más pequeño tal que el 50% de la longitud del ensamblaje está contenido en *contigs* más largos o iguales que x^2 . Esta estadística está destinada a medir la contigüidad del ensamblaje. Generalmente se prefiere que esta estadística sea tan larga como sea posible. Dado que depende de la longitud del ensamblaje, esta estadística puede ser ambigua cuando se desea comparar la calidad entre distintos ensamblajes con diferentes longitudes, pues no hay una referencia común de comparación.
- **NG50**: análoga al N50, esta estadística está calculada con base en un genoma de referencia para solucionar los problemas del N50 al comparar varios ensamblajes (Earl et al. 2011). Es la longitud x del *contig* más pequeño tal que el 50% de la longitud del **genoma de referencia** está contenido en *contigs* más largos o iguales que x^3 .
- **L50**: el número de *contigs* con longitud mayor o igual a N50⁴.
- **LG50**: análogo al L50 pero con base en el genoma de referencia. El número de *contigs* con longitud mayor o igual a NG50.

Desde el punto de vista del número de *contigs*, el mejor ensamblaje es el [SRR18335438](#), dado que tiene 50 *contigs*, el menor número entre las cuatro muestras. Por el contrario, el ensamblaje [SRR18335439](#) tiene 269, un número muy elevado para un ensamblaje. Si observamos la longitud total de los ensamblajes, todos tienen un tamaño cercano al tamaño del genoma de *H. pylori* de 1.6 Mbp, excepto por el [SRR18335439](#) que tiene 5.8 Mbp, lo cual corresponde con la altísima, y nunca deseada, cantidad de *contigs* obtenidos. De nuevo

²Wikipedia - Estadístico N50: https://es.wikipedia.org/wiki/Estad%C3%ADstico_N50.

³Wikipedia - Estadístico N50, Estadístico NG50: https://es.wikipedia.org/wiki/Estad%C3%ADstico_N50#Estad%C3%ADstico_NG50.

⁴Wikipedia - Estadístico N50, Estadístico L50: https://es.wikipedia.org/wiki/Estad%C3%ADstico_N50#Estad%C3%ADstico_L50.

5. Control de calidad de ensamblajes

SRR18335438 es el mejor ensamblaje pues es el más cercano al tamaño esperado. En la figura Figura 5.2 se observa el gráfico de longitud acumulada de los *contigs* por cada ensamblaje. Se puede corroborar que el ensamblaje SRR18335439 (línea azul) está muy por arriba del tamaño esperado del genoma de referencia (línea negra punteada).

Claramente el ensamblaje SRR18335439 no está del todo bien, de acuerdo con el número de *contigs* y su longitud. Haber incluido esta muestra en este manual fue intencional; se deseaba mostrar un ejemplo de un ensamblaje que no resulta bien a pesar de seguir los controles de calidad mínimos requeridos sobre los datos. Debido a la baja calidad de este ensamblaje, no se considerará esta muestra en el siguiente paso de anotación funcional.

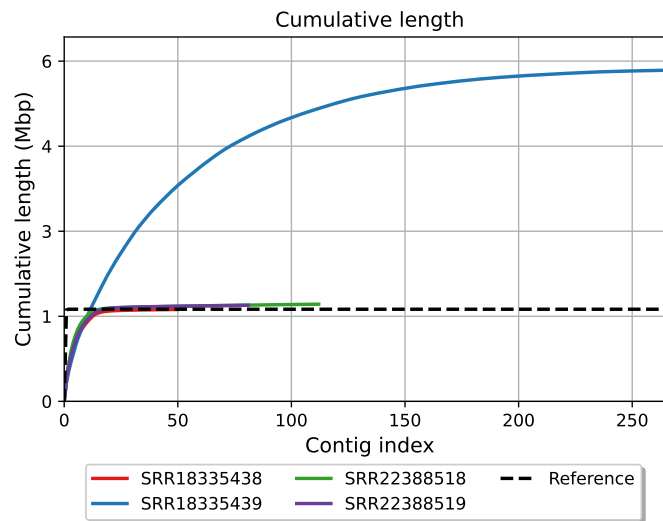


Figura 5.2.: Gráfico de longitud acumulada de los ensamblajes

5.4. Corrección de ensamblajes

La baja calidad del ensamblaje SRR18335439 puede ser debida a la presencia de contaminantes en la librería de ADN preparada para secuenciación. Dichos contaminantes pueden corresponder a ADN de otro u otros organismos, u otras cepas de *H. pylori* distintas (y más divergentes) de la inicialmente aislada.

Existen varias opciones para corregir o mejorar un ensamblaje, las cuales incluyen la descontaminación de los datos antes del ensamblaje, descontaminación del ensamblaje, y ensamblaje de los *contigs* o *scaffolds* obtenidos. Estas opciones son ejecutadas con otras herramientas bioinformáticas especializadas para cada caso. Sin embargo, no se tratarán en este documento (¡al menos no ahora!) pues, como se dijo al inicio, la intención del

5. *Control de calidad de ensamblajes*

manual es mostrar el caso más sencillo de ensamblaje y anotación de genomas bacterianos con los pasos más básicos. Es posible que en un futuro se incluyan secciones en el manual para la corrección y mejoramiento de los ensamblajes.

6. Anotación

Hemos llegado al último paso de este manual, la anotación funcional de los ensamblajes. La finalidad de este paso consiste en “dar identidad” a las secuencias de los ensamblajes. Brevemente, la anotación consiste en identificar características en las secuencias de *contigs/scaffolds*, como por ejemplo genes, encontrar información útil a cerca de ellas y finalmente etiquetar o anotar las características con esta información. Entre la información que se busca, la función biológica de los genes es una de las más importantes. La información se encuentra por medio de búsquedas en bases de datos públicas especializadas, basadas en similitud. Haciendo uso de la homología de secuencia, si la secuencia *A* de la cual se desconoce su función, es lo suficientemente similar (usualmente $sim \geq 80\%$) a la secuencia *B* previamente caracterizada, entonces se puede asociar la función de *B* a la secuencia *A*.

6.1. Software requerido

Software	Versión	Descripción	Ref.
Prokka	1.14.6	Herramienta de software para la anotación rápida de genomas de bacterias, virus y arqueas.	Seemann (2014); Repositorio GitHub

6.2. Ejecución de Prokka

Anotaremos los ensamblajes usando la herramienta Prokka, la cual consiste en un *pipeline* que involucra varias herramientas bioinformáticas para identificar las características, hacer las búsquedas en las bases de datos biológicas, y finalmente etiquetar las secuencias de las características identificadas.

Para anotar el ensamblaje [SRR22388518](#), ejecute las siguientes órdenes:

6. Anotación

```
# Entrar en el directorio raíz del proyecto
cd $HOME/proyectos/ensamblaje

# Ejecutar la anotación
prokka \
  --cpus 4 \
  --genus "Helicobacter" \
  --species "pylori" \
  --locustag "SRR22388518" \
  --outdir resultados/03_annotacion/SRR22388518 \
  --prefix "SRR22388518"\
  resultados/01_ensamblaje/SRR22388518.fasta
```

Veamos los argumentos usados para la anotación con Prokka:

- **cpus**: el número de *cores* (cpus) a usar por Prokka para agilizar el proceso.
- **genus**: el género del organismo del ensamblaje que queremos anotar.
- **species**: la especie del organismo del ensamblaje que queremos anotar.
- **locustag**: la etiqueta que se usa para agregar a todas las características encontradas y anotadas por Prokka para este ensamblaje. En este caso se usó el *accession number* de la muestra.
- **outdir**: el directorio en donde se guardarán todos los archivos de salida de Prokka.
- **prefix**: el prefijo (nombre) para los diferentes archivos de salida de Prokka. El prefijo siempre será el mismo pero cambia la extensión de cada archivo. En este caso se usó igualmente el *accession number*.
- **resultados/01_ensamblaje/SRR22388518.fasta**: el archivo de entrada (ensamblaje) a ser anotado.

Al ejecutar el comando verá una salida parecida a esta:

```
[19:13:29] This is prokka 1.14.6
[19:13:29] Written by Torsten Seemann <torsten.seemann@gmail.com>
[19:13:29] Homepage is https://github.com/tseemann/prokka
[19:13:29] Local time is Fri Mar 24 19:13:29 2023
[19:13:29] You are hector
[19:13:29] Operating system is linux
[19:13:29] You have BioPerl 1.7.8
Argument 1.7.8 isn't numeric in numeric lt (<) at
  ↪ /home/hector/mambaforge/envs/ensam/bin/prokka line 259.
[19:13:29] System has 8 cores.
[19:13:29] Will use maximum of 4 cores.
```

6. Anotación

```
[19:13:29] Annotating as >>> Bacteria <<<
[19:13:29] Creating new output folder:
  ↳ resultados/03_annotacion/SRR22388518
[19:13:29] Running: mkdir -p resultados\03_annotacion\03_annotacion\SRR22388518
[19:13:29] Using filename prefix: SRR22388518.XXX
[19:13:29] Setting HMMER_NCPU=1
[19:13:29] Writing log to:
  ↳ resultados/03_annotacion/SRR22388518/SRR22388518.log
[19:13:29] Command: /home/hector/mambaforge/envs/ensam/bin/prokka
  ↳ --cpus 4 --genus Helicobacter --species pylori --locustag
  ↳ SRR22388518 --outdir resultados/03_annotacion/SRR22388518 --prefix
  ↳ SRR22388518 resultados/01_ensamblaje/SRR22388518.fasta

...

[19:14:30] Output files:
[19:14:30] resultados/03_annotacion/SRR22388518/SRR22388518.fsa
[19:14:30] resultados/03_annotacion/SRR22388518/SRR22388518.fna
[19:14:30] resultados/03_annotacion/SRR22388518/SRR22388518.gff
[19:14:30] resultados/03_annotacion/SRR22388518/SRR22388518.sqn
[19:14:30] resultados/03_annotacion/SRR22388518/SRR22388518.tbl
[19:14:30] resultados/03_annotacion/SRR22388518/SRR22388518.gbk
[19:14:30] resultados/03_annotacion/SRR22388518/SRR22388518.ffn
[19:14:30] resultados/03_annotacion/SRR22388518/SRR22388518.err
[19:14:30] resultados/03_annotacion/SRR22388518/SRR22388518.faa
[19:14:30] resultados/03_annotacion/SRR22388518/SRR22388518.tsv
[19:14:30] resultados/03_annotacion/SRR22388518/SRR22388518.txt
[19:14:30] resultados/03_annotacion/SRR22388518/SRR22388518.log
[19:14:30] Annotation finished successfully.
[19:14:30] Walltime used: 1.02 minutes
[19:14:30] If you use this result please cite the Prokka paper:
[19:14:30] Seemann T (2014) Prokka: rapid prokaryotic genome annotation.
  ↳ Bioinformatics. 30(14):2068-9.
[19:14:30] Type 'prokka --citation' for more details.
[19:14:30] Share and enjoy!
```

La salida de Prokka es bastante detallada y extensa. La primera parte muestra un resumen de las opciones pasadas al programa, incluido el comando ejecutado y la ubicación del archivo *log* (*Writing log to: ...*) con toda la información de salida, muy útil para depuración de errores en caso de que se presenten problemas al ejecutar Prokka. En la parte final se encuentra la lista de archivos de salida, los cuales están ubicados en el

6. Anotación

directorio `resultados/03_annotacion`. En la Tabla 6.2 se presenta la descripción de cada archivo de salida.

Tabla 6.2.: Descripción de los archivos de salida generados por Prokka.

Extensión	Descripción
<code>.gff</code>	El archivo de anotación principal en formato GFF3 que contiene las secuencias y las anotaciones.
<code>.gbk</code>	Archivo en formato Genbank derivado del archivo principal <code>.gff</code> .
<code>.fna</code>	Archivo de nucleótidos en formato FASTA de las secuencias de los <i>contigs/scaffolds</i> de entrada.
<code>.faa</code>	Archivo de proteínas en formato FASTA de las traducciones de las secuencias codificantes CDS (del inglés <i>coding sequences</i>).
<code>.ffn</code>	Archivo de nucleótidos en formato FASTA de todos los transcritos predichos (CDS, rRNA, tRNA, tmRNA, misc_RNA).
<code>.sqn</code>	Un archivo "Seqin" en formato ASN1 para envío a la base de datos Genbank, en caso de ser requerido.
<code>.fsa</code>	Archivo de nucleótidos en formato FASTA de las secuencias de los <i>contigs/scaffolds</i> de entrada, usado por el programa <code>tbl2asn</code> para crear el archivo <code>.sqn</code> . Es casi el mismo archivo <code>.fna</code> pero con etiquetas adicionales Seqin en los identificadores de las secuencias.
<code>.tbl</code>	Archivo de tabla de características usado por <code>tbl2asn</code> para crear el archivo <code>.sqn</code> .
<code>.err</code>	Archivo de reporte de discrepancia para NCBI, en caso de ser requerido, con las anotaciones no aceptables.
<code>.log</code>	Archivo con toda la salida generada por Prokka durante la ejecución del comando.
<code>.txt</code>	Estadísticas de las características encontradas y anotadas.
<code>.tsv</code>	Archivo de texto separado por tabulaciones (TSV, del inglés <i>text separated values</i>) con todas las características: <code>locus_tag</code> , <code>ftype</code> , <code>len_bp</code> , <code>gene</code> , <code>EC_number</code> , <code>COG</code> , <code>product</code> .

Ahora podemos anotar el resto de los ensamblajes:

```
prokka \  
  --cpus 4 \  
  --genus "Helicobacter" \  
  --species "pylori" \  
  --
```

6. Anotación

```
--locustag "SRR22388519" \  
--outdir resultados/03_annotacion/SRR22388519 \  
--prefix "SRR22388519"\  
resultados/01_ensamblaje/SRR22388519.fasta  
  
prokka \  
--cpus 4 \  
--genus "Helicobacter" \  
--species "pylori" \  
--locustag "SRR18335438" \  
--outdir resultados/03_annotacion/SRR18335438 \  
--prefix "SRR18335438"\  
resultados/01_ensamblaje/SRR18335438.fasta
```

El archivo más importante generado por Prokka es el archivo principal `.gff`, el cual puede ser visualizado con el software IGV¹ (Robinson et al. (2011); del inglés *Integrative Genomics Viewer*).

Finalmente, los archivos de anotación son requeridos para posteriores análisis de genómica comparativa, por ejemplo para la identificación del pangenoma² de un grupo de genomas, el cual suele ser utilizado para identificación de genes marcadores para especies, serotipos³, o cepas específicas de organismos bacterianos. Estos análisis de genómica comparativa son usados en epidemiología basada en secuenciación, la cual es fundamental en salud pública para la vigilancia y seguimiento de brotes infecciosos.

¹*Integrative Genomics Viewer*: <http://software.broadinstitute.org/software/igv/home>

²Wikipedia - Pangenoma: <https://es.wikipedia.org/wiki/Pangenoma>.

³Wikipedia - Serotipo: <https://es.wikipedia.org/wiki/Serotipo>.

Bibliografía

- Chen, Shifu, Yanqing Zhou, Yaru Chen, y Jia Gu. 2018. «fastp: an ultra-fast all-in-one FASTQ preprocessor». *Bioinformatics* 34 (17): i884-90. <https://doi.org/10.1093/bioinformatics/bty560>.
- Cock, Peter J. A., Christopher J. Fields, Naohisa Goto, Michael L. Heuer, y Peter M. Rice. 2010. «The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants». *Nucleic Acids Research* 38 (6): 1767-71. <https://doi.org/10.1093/nar/gkp1137>.
- Compeau, Phillip E. C., Pavel A. Pevzner, y Glenn Tesler. 2011. «Why are de Bruijn graphs useful for genome assembly?» *Nature biotechnology* 29 (11): 987-91. <https://doi.org/10.1038/nbt.2023>.
- Earl, Dent, Keith Bradnam, John St John, Aaron Darling, Dawei Lin, Joseph Fass, Hung On Ken Yu, et al. 2011. «Assemblathon 1: A Competitive Assessment of de Novo Short Read Assembly Methods». *Genome Research* 21 (12): 2224-41. <https://doi.org/10.1101/gr.126599.111>.
- Espitia-Navarro, Hector F, Aroon T Chande, Shashwat D Nagar, Heather Smith, I King Jordan, y Lavanya Rishishwar. 2020. «STing: accurate and ultrafast genomic profiling with exact sequence matches». *Nucleic Acids Research* 48 (14): 7681-89. <https://doi.org/10.1093/nar/gkaa566>.
- Ewels, Philip, Måns Magnusson, Sverker Lundin, y Max Käller. 2016. «MultiQC: summarize analysis results for multiple tools and samples in a single report». *Bioinformatics* 32 (19): 3047-48. <https://doi.org/10.1093/bioinformatics/btw354>.
- Gurevich, Alexey, Vladislav Saveliev, Nikolay Vyahhi, y Glenn Tesler. 2013. «QUAST: quality assessment tool for genome assemblies». *Bioinformatics* 29 (8): 1072-75. <https://doi.org/10.1093/bioinformatics/btt086>.
- Katz, Kenneth, Oleg Shutov, Richard Lapoint, Michael Kimelman, J Rodney Brister, y Christopher O'Sullivan. 2022. «The Sequence Read Archive: a decade more of explosive growth». *Nucleic Acids Research* 50 (D1): D387-90. <https://doi.org/10.1093/nar/gkab1053>.
- Kodaman, Nuri, Alvaro Pazos, Barbara G. Schneider, M. Blanca Piazuelo, Robertino Mera, Rafal S. Sobota, Liviu A. Sicinschi, et al. 2014. «Human and Helicobacter pylori coevolution shapes the risk of gastric disease». *Proceedings of the National Academy of Sciences* 111 (4): 1455-60. <https://doi.org/10.1073/pnas.1318093111>.
- Prjibelski, Andrey, Dmitry Antipov, Dmitry Meleshko, Alla Lapidus, y Anton

Bibliografía

- Korobeynikov. 2020. «Using SPAdes De Novo Assembler». *Current Protocols in Bioinformatics* 70 (1): e102. <https://doi.org/10.1002/cpbi.102>.
- Robinson, James T., Helga Thorvaldsdóttir, Wendy Winckler, Mitchell Guttman, Eric S. Lander, Gad Getz, y Jill P. Mesirov. 2011. «Integrative Genomics Viewer». *Nature biotechnology* 29 (1): 24-26. <https://doi.org/10.1038/nbt.1754>.
- Seemann, Torsten. 2014. «Prokka: rapid prokaryotic genome annotation». *Bioinformatics* 30 (14): 2068-69. <https://doi.org/10.1093/bioinformatics/btu153>.
- Wick, Ryan R., Mark B. Schultz, Justin Zobel, y Kathryn E. Holt. 2015. «Bandage: interactive visualization of de novo genome assemblies». *Bioinformatics* 31 (20): 3350-52. <https://doi.org/10.1093/bioinformatics/btv383>.

Acerca de los autores

Héctor Fabio Espitia-Navarro

Héctor Fabio Espitia-Navarro es Ph.D. en bioinformática de *Georgia Institute of Technology* (Atlanta, Georgia, EE. UU.) y actualmente se desempeña como científico de datos (*Data Scientist*) especializado en bioinformática, en el *Corporate Reserach Materials Laboratory* de la compañía 3M (Saint Paul, Minnesota, EE. UU.). Obtuvo su título de Ingeniero en Sistemas y un Máster en Ingeniería en la Universidad del Valle (Cali, Valle, Colombia). Es miembro del Grupo de Investigación Salud Pública de la Universidad de Nariño⁴ (Pasto, Nariño, Colombia), y del Grupo de Investigación de Bioinformática y Biocomputación de la Universidad del Valle⁵ (Cali, Valle, Colombia). Además, es colaborador del Centro de Investigación de la Caña de Azúcar de Colombia - Cenicaña⁶ (Cali, Valle, Colombia) y del *Indian River Research Education Center*⁷ de *Florida University* (Fort Pierce, Florida, EE. UU.).

El Dr. Espitia-Navarro se especializa en investigación en bioinformática aplicada, especialmente enfocada en la genómica comparativa y la metagenómica, y se interesa por el desarrollo de software y algoritmos bioinformáticos. Empezó su trabajo en bioinformática desde el pregrado con el desarrollo de una aplicación para análisis multifractal de secuencias de ADN y proteínas. Posteriormente, durante su maestría trabajó en Cenicaña en la integración de la bioinformática para el mejoramiento del cultivo de la caña de azúcar. Específicamente, desarrolló un modelo computacional para la identificación y *ranking* de genes candidatos relacionados con la tolerancia a estrés hídrico en caña de azúcar. Después, durante



Universidad de **Nariño**
FUNDADA EN 1904



C E S U N
CENTRO DE ESTUDIOS EN SALUD
Universidad de Nariño

⁴Grupo de Investigación Salud Pública, GrupLac, Minciencias, Colombia: <https://scienti.minciencias.gov.co/gruplac/jsp/visualiza/visualizagr.jsp?nro=00000000008168>

⁵Grupo de Investigación de Bioinformática y Biocomputación, GrupLac, Minciencias, Colombia: <https://scienti.minciencias.gov.co/gruplac/jsp/visualiza/visualizagr.jsp?nro=00000000000759>

⁶Cenicaña, sitio Web: <https://www.cenicana.org/>

⁷IRREC, sitio Web: <https://irrec.ifas.ufl.edu/>

su doctorado trabajó en el desarrollo de algoritmos para epidemiología molecular basada en secuenciación de segunda generación (NGS del inglés *Next Generation Sequencing*). Su principal proyecto fue STing (Espitia-Navarro et al. 2020), un algoritmo libre de alineamiento de secuencias basado en frecuencias de *k-mers*, para la eficiente caracterización y clasificación taxonómica de patógenos bacterianos. Actualmente su trabajo en la compañía 3M está orientado al uso la bioinformática y el aprendizaje automático (*machine learning*) en la investigación en los campos de la purificación de biofarmacéuticos y la cicatrización de heridas.

Alvaro Pazos Moncayo

Alvaro Pazos Moncayo nació en Pasto, ciudad capital del departamento de Nariño, Colombia. Es profesor titular de la Universidad de Nariño desde hace más de 30 años. Ingresó a la edad de 15 años a la Universidad de Antioquia donde cursó sus primeros estudios en Ingeniería Química. Gran parte de su formación académica la realizó en la Universidad del Valle (Cali, Valle, Colombia), donde obtuvo el título de pregrado en Bacteriología y Laboratorio Clínico y una maestría en Microbiología. Cursó estudios de doctorado en Ciencias Biomédicas en la Universidad del Valle y en *Vanderbilt University* (Nashville, Tennessee, EE. UU.) desarrolló su tesis doctoral, bajo la orientación del ilustre científico colombiano Dr. Pelayo Correa. Hizo un posdoctorado en metagenómica clínica en el Laboratorio de Genómica Computacional del Dr. King Jordan, en *Georgia Institute of Technology* (Atlanta, Georgia, EE. UU.). Su posicionamiento en investigación le permitió gestionar, como Director y junto con los investigadores del Grupo de Investigación Salud Pública de la Universidad de Nariño, su reconocimiento por la convocatoria No. 894 de 2021 de MinCiencias (Colombia) la máxima clasificación A1 para el grupo⁸. En la misma convocatoria de reconocimiento de investigadores obtuvo la categoría de Investigador Emérito con vigencia vitalicia⁹ y el premio Ángela Restrepo Moreno a la Excelencia en Ciencia, Tecnología e Innovación de MinCiencias 2022. Su producción intelectual de más de 35 artículos científicos publicados en revistas extranjeras en el idioma inglés, sus proyectos de investigación, su reconocimiento nacional e internacional con 8 premios y su trayectoria académica se plasman en su CVLac¹⁰; en el *Google Scholar*¹¹; en el *Research*

⁸Grupo de Investigación Salud Pública, GrupLac, Minciencias, Colombia: <https://scienti.minciencias.gov.co/gruplac/jsp/visualiza/visualizagr.jsp?nro=00000000008168>

⁹Alvaro Pazos, CVLac, Minciencias, Colombia: https://scienti.minciencias.gov.co/cvlac/visualizador/generarCurriculoCv.do?cod_rh=0000194972

¹⁰Alvaro Pazos, CVLac, Minciencias, Colombia: https://scienti.minciencias.gov.co/cvlac/visualizador/generarCurriculoCv.do?cod_rh=0000194972

¹¹Alvaro Pazos, *Google Scholar*: <https://scholar.google.com/citations?user=n4ZJz4cAAAAJ&hl=es>

Gate¹² y el *Open Research and Contributor ID (ORCID)*¹³. Su campo de actuación científica se enfoca en el estudio de las relaciones biológicas entre el onco patógeno *Helicobacter pylori* y el humano. Sus perspectivas de trabajo se centran en la aplicación de la teoría evolutiva en la prevención del cáncer gástrico en Nariño (Colombia). Su aspiración basada en evidencia científica con publicación en una de las revistas mas prestigiosas en ciencias de los Estados Unidos, *Proceedings of the National Academy of Sciences (PNAS)* (Kodaman et al. 2014) , es consolidar las bases de la relaciones coevolutivas entre humano y *Helicobacter pylori* como determinantes importantes del riesgo de enfermedad gástrica y que la colonización histórica de las Américas sigue influyendo en la salud de las poblaciones americanas modernas. Los aportes de la genómica clínica en especial el manual “Ensamblaje y Anotación de Genomas Bacterianos” con aplicación en investigación traslacional, sin duda impactarán en la toma de decisiones en políticas públicas que propendan por establecer programas de prevención de enfermedades prioritarias en salud en Colombia.

¹² Alvaro Pazos, *Research Gate*: <https://www.researchgate.net/profile/Alvaro-Pazos>

¹³ Alvaro Pazos, *Open Research and Contributor ID (ORCID)*: <https://orcid.org/0000-0001-5603-7898>

A. FastQC help

i Nota

Esta es una copia de la ayuda de FastQC (en inglés) disponible en <https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/>

A.1. Introduction

A.1.1. What is FastQC

Modern high throughput sequencers can generate hundreds of millions of sequences in a single run. Before analysing this sequence to draw biological conclusions you should always perform some simple quality control checks to ensure that the raw data looks good and there are no problems or biases in your data which may affect how you can usefully use it.

Most sequencers will generate a QC report as part of their analysis pipeline, but this is usually only focused on identifying problems which were generated by the sequencer itself. FastQC aims to provide a QC report which can spot problems which originate either in the sequencer or in the starting library material.

FastQC can be run in one of two modes. It can either run as a stand alone interactive application for the immediate analysis of small numbers of FastQ files, or it can be run in a non-interactive mode where it would be suitable for integrating into a larger analysis pipeline for the systematic processing of large numbers of files.

A.2. Basic Operations

A.2.1. Opening a Sequence file

To open one or more Sequence files interactively simply run the program and select File > Open. You can then select the files you want to analyse.

Newly opened files will immediately appear in the set of tabs at the top of the screen. Because of the size of these files it can take a couple of minutes to open them. FastQC operates a queueing system where only one file is opened at a time, and new files will wait until existing files have been processed.

FastQC supports files in the following formats

- FastQ (all quality encoding variants)
- Casava FastQ files¹
- Colospace FastQ
- GZip compressed FastQ
- SAM
- BAM
- SAM/BAM Mapped only (normally used for colospace data)

By default FastQC will try to guess the file format from the name of the input file. Anything ending in .sam or .bam will be opened as a SAM/BAM file (using all sequences, mapped and unmapped) , and everything else will be treated as FastQ format. If you want to override this detection and specify the file format manually then you can use the drop down file filter in the file chooser to select the type of file you're going to load. You need to use the drop down selector to make the program use the Mapped BAM or Casava file modes as these won't be selected automatically.

A.2.2. Evaluating Results

The analysis in FastQC is performed by a series of analysis modules. The left hand side of the main interactive display or the top of the HTML report show a summary of the modules which were run, and a quick evaluation of whether the results of the module seem entirely normal (green tick), slightly abnormal (orange triangle) or very unusual (red cross).

¹Casava fastq format is the same as regular fastq except that the data is usually split across multiple files for a single sample. In this mode the program will merge the files in a sample group and present a single report for each sample. Also Casava fastq files contain poor quality sequences which have been flagged to be remove. In Casava mode the program will exclude these flagged sequences from the report.

A. *FastQC help*

It is important to stress that although the analysis results appear to give a pass/fail result, these evaluations must be taken in the context of what you expect from your library. A 'normal' sample as far as FastQC is concerned is random and diverse. Some experiments may be expected to produce libraries which are biased in particular ways. You should treat the summary evaluations therefore as pointers to where you should concentrate your attention and understand why your library may not look random and diverse.

Specific guidance on how to interpret the output of each module can be found in the modules section of the help.

A.2.3. Saving a Report

In addition to providing an interactive report FastQC also has the option to create an HTML version of this report for a more permanent record. This HTML report can also be generated directly by running FastQC in non-interactive mode.

To create a report simply select File > Save Report from the main menu. By default a report will be created using the name of the fastq file with `_fastqc.html` appended to the end. The report will be created for whichever file tab was active when the menu option was selected.

The HTML file which is saved is a self-contained document with all of the graphs embedded into it, so you can distribute this single file. Alongside the HTML file is a zip file (with the same name as the HTML file, but with `.zip` added to the end). This file contains the graphs from the report as separate files but also contains data files which are designed to be easily parsed to allow for a more detailed and automated evaluation of the raw data on which the QC report is built.

A.3. Analysis Modules

A.3.1. Basic Statistics

A.3.1.1. Summary

The Basic Statistics module generates some simple composition statistics for the file analysed.

- **Filename:** The original filename of the file which was analysed
- **File type:** Says whether the file appeared to contain actual base calls or colorspace data which had to be converted to base calls

A. *FastQC help*

- **Encoding:** Says which ASCII encoding of quality values was found in this file.
- **Total Sequences:** A count of the total number of sequences processed. There are two values reported, actual and estimated. At the moment these will always be the same. In the future it may be possible to analyse just a subset of sequences and estimate the total number, to speed up the analysis, but since we have found that problematic sequences are not evenly distributed through a file we have disabled this for now.
- **Filtered Sequences:** If running in Casava mode sequences flagged to be filtered will be removed from all analyses. The number of such sequences removed will be reported here. The total sequences count above will not include these filtered sequences and will be the number of sequences actually used for the rest of the analysis.
- **Sequence Length:** Provides the length of the shortest and longest sequence in the set. If all sequences are the same length only one value is reported.
- **%GC:** The overall %GC of all bases in all sequences

A.3.1.2. Warning

Basic Statistics never raises a warning.

A.3.1.3. Failure

Basic Statistics never raises an error.

A.3.1.4. Common reasons for warnings

This module never raises warnings or errors

A.3.2. Adapter Content

A.3.2.1. Summary

The Kmer Content module will do a generic analysis of all of the Kmers in your library to find those which do not have even coverage through the length of your reads. This can find a number of different sources of bias in the library which can include the presence of read-through adapter sequences building up on the end of your sequences.

You can however find that the presence of any overrepresented sequences in your library (such as adapter dimers) will cause the Kmer plot to be dominated by the Kmers these

A. *FastQC help*

sequences contain, and that it's not always easy to see if there are other biases present in which you might be interested.

One obvious class of sequences which you might want to analyse are adapter sequences. It is useful to know if your library contains a significant amount of adapter in order to be able to assess whether you need to adapter trim or not. Although the Kmer analysis can theoretically spot this kind of contamination it isn't always clear. This module therefore does a specific search for a set of separately defined Kmers and will give you a view of the total proportion of your library which contain these Kmers. A results trace will always be generated for all of the sequences present in the adapter config file so you can see the adapter content of your library, even if it's low.

The plot itself shows a cumulative percentage count of the proportion of your library which has seen each of the adapter sequences at each position. Once a sequence has been seen in a read it is counted as being present right through to the end of the read so the percentages you see will only increase as the read length goes on.

In addition to classic adapter sequences the default configuration also includes polyA and polyG as sequences to search for. PolyA can be useful to include when looking at RNA-Seq libraries. PolyG is present as a technical artefact in 2-colour illumina libraries where it is produced when the signal from the cluster disappears. Both of these sequences are generally trimmed from the 3' end of sequences, and are therefore removed in a similar way to adapters, hence their inclusion in the default configuration. These sequences can be removed by editing the `adapter_list.txt` file in the Configuration directory.

A.3.2.2. Warning

This module will issue a warning if any sequence is present in more than 5% of all reads.

A.3.2.3. Failure

This module will issue a warning if any sequence is present in more than 10% of all reads.

A.3.2.4. Common reasons for warnings

Any library where a reasonable proportion of the insert sizes are shorter than the read length will trigger this module. This doesn't indicate a problem as such - just that the sequences will need to be adapter trimmed before proceeding with any downstream analysis.

A.3.3. Kmer Content

A.3.3.1. Summary

The analysis of overrepresented sequences will spot an increase in any exactly duplicated sequences, but there are a different subset of problems where it will not work.

- If you have very long sequences with poor sequence quality then random sequencing errors will dramatically reduce the counts for exactly duplicated sequences.
- If you have a partial sequence which is appearing at a variety of places within your sequence then this won't be seen either by the per base content plot or the duplicate sequence analysis.

The Kmer module starts from the assumption that any small fragment of sequence should not have a positional bias in its appearance within a diverse library. There may be biological reasons why certain Kmers are enriched or depleted overall, but these biases should affect all positions within a sequence equally. This module therefore measures the number of each 7-mer at each position in your library and then uses a binomial test to look for significant deviations from an even coverage at all positions. Any Kmers with positionally biased enrichment are reported. The top 6 most biased Kmer are additionally plotted to show their distribution.

To allow this module to run in a reasonable time only 2% of the whole library is analysed and the results are extrapolated to the rest of the library. Sequences longer than 500bp are truncated to 500bp for this analysis.

A.3.3.2. Warning

This module will issue a warning if any k-mer is imbalanced with a binomial p-value <0.01.

A.3.3.3. Failure

This module will issue a warning if any k-mer is imbalanced with a binomial p-value < 10^{-5} .

A. FastQC help

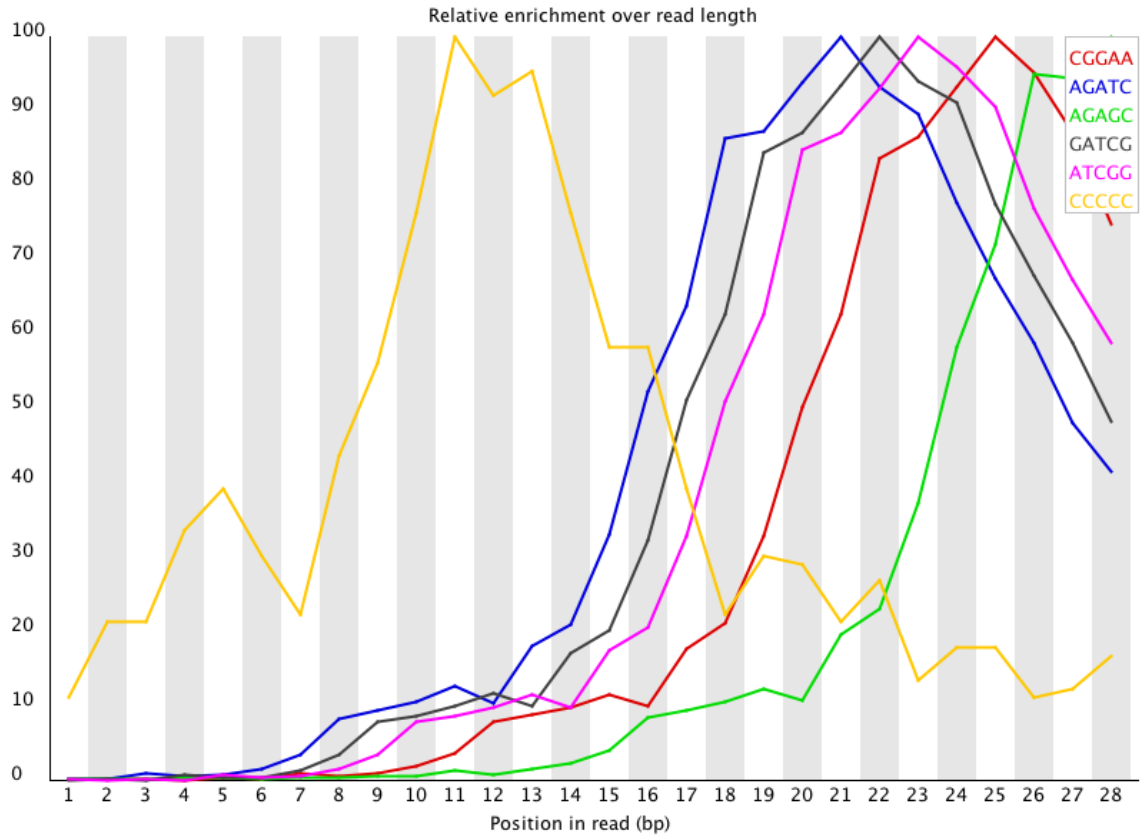


Figura A.1.: Kmer profiles

A. *FastQC help*

A.3.3.4. Common reasons for warnings

Any individually overrepresented sequences, even if not present at a high enough threshold to trigger the overrepresented sequences module will cause the Kmers from those sequences to be highly enriched in this module. These will normally appear as sharp spikes of enrichment at a single point in the sequence, rather than a progressive or broad enrichment.

Libraries which derive from random priming will nearly always show Kmer bias at the start of the library due to an incomplete sampling of the possible random primers.

A.3.4. Per Tile Sequence Quality

A.3.4.1. Summary

This graph will only appear in your analysis results if you're using an Illumina library which retains its original sequence identifiers. Encoded in these is the flowcell tile from which each read came. The graph allows you to look at the quality scores from each tile across all of your bases to see if there was a loss in quality associated with only one part of the flowcell.

The plot shows the deviation from the average quality for each tile. The colours are on a cold to hot scale, with cold colours being positions where the quality was at or above the average for that base in the run, and hotter colours indicate that a tile had worse qualities than other tiles for that base. In the example below you can see that certain tiles show consistently poor quality. A good plot should be blue all over.

Reasons for seeing warnings or errors on this plot could be transient problems such as bubbles going through the flowcell, or they could be more permanent problems such as smudges on the flowcell or debris inside the flowcell lane.

A.3.4.2. Warning

This module will issue a warning if any tile shows a mean Phred score more than 2 less than the mean for that base across all tiles.

A.3.4.3. Failure

This module will raise an error if any tile shows a mean Phred score more than 5 less than the mean for that base across all tiles.

A. *FastQC help*

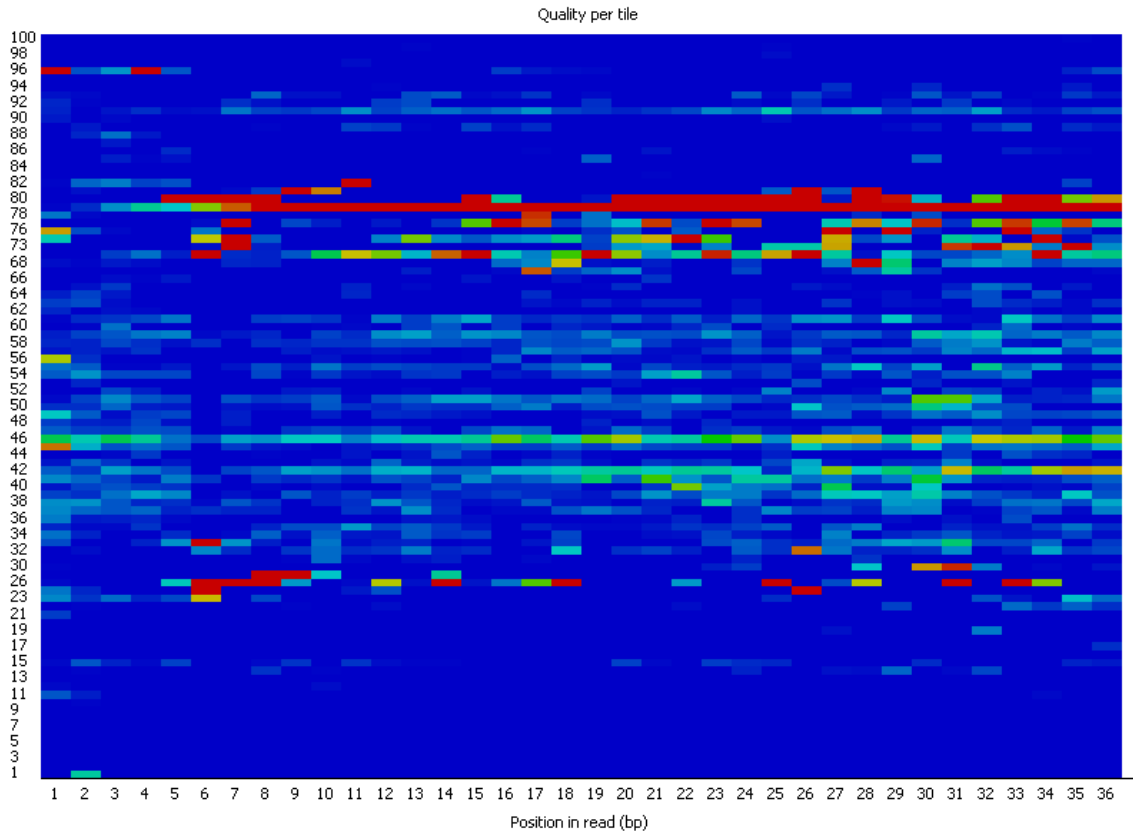


Figura A.2.: Kmer profiles

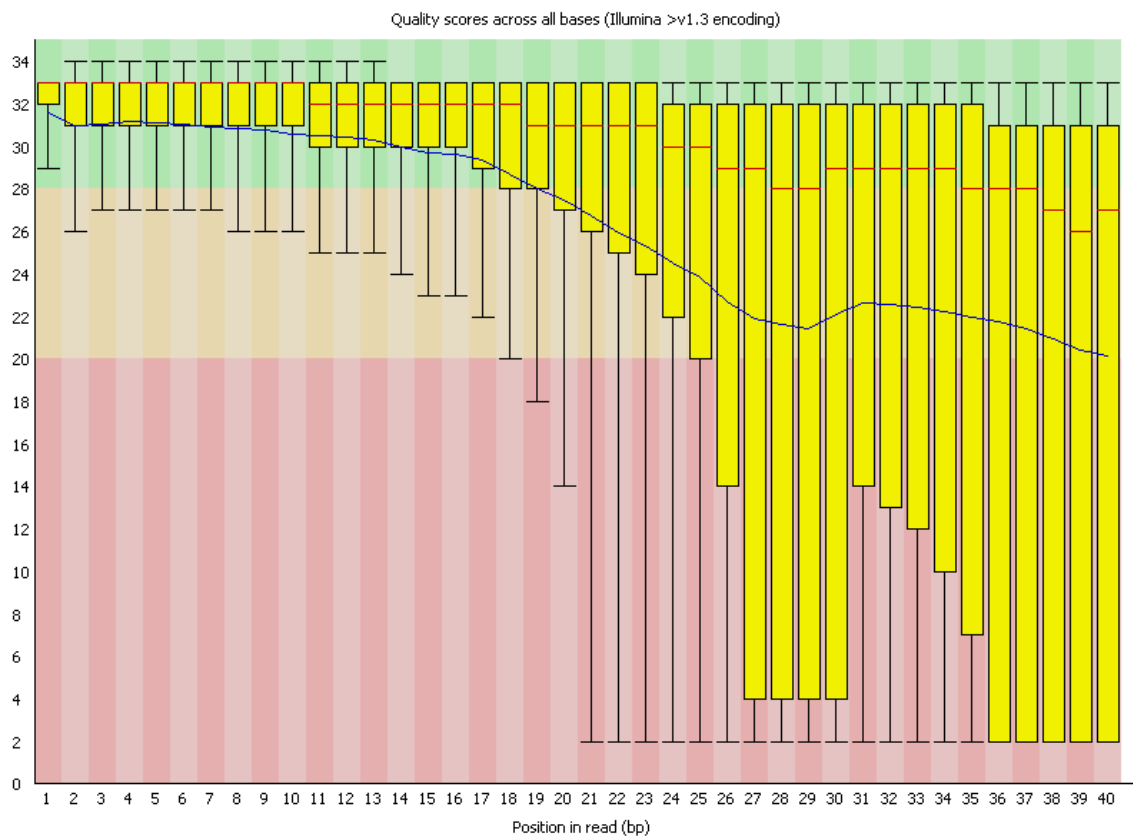
A.3.4.4. Common reasons for warnings

Whilst warnings in this module can be triggered by individual specific events we have also observed that greater variation in the phred scores attributed to tiles can also appear when a flowcell is generally overloaded. In this case events appear all over the flowcell rather than being confined to a specific area or range of cycles. We would generally ignore errors which mildly affected a small number of tiles for only 1 or 2 cycles, but would pursue larger effects which showed high deviation in scores, or which persisted for several cycles.

A.3.5. Per Base Sequence Quality

A.3.5.1. Summary

This view shows an overview of the range of quality values across all bases at each position in the FastQ file.



A. *FastQC help*

For each position a BoxWhisker type plot is drawn. The elements of the plot are as follows:

- The central red line is the median value
- The yellow box represents the inter-quartile range (25-75%)
- The upper and lower whiskers represent the 10% and 90% points
- The blue line represents the mean quality

The y-axis on the graph shows the quality scores. The higher the score the better the base call. The background of the graph divides the y axis into very good quality calls (green), calls of reasonable quality (orange), and calls of poor quality (red). The quality of calls on most platforms will degrade as the run progresses, so it is common to see base calls falling into the orange area towards the end of a read.

It should be mentioned that there are number of different ways to encode a quality score in a FastQ file. FastQC attempts to automatically determine which encoding method was used, but in some very limited datasets it is possible that it will guess this incorrectly (ironically only when your data is universally very good!). The title of the graph will describe the encoding FastQC thinks your file used.

Results from this module will not be displayed if your input is a BAM/SAM file in which quality scores have not been recorded.

A.3.5.2. Warning

A warning will be issued if the lower quartile for any base is less than 10, or if the median for any base is less than 25.

A.3.5.3. Failure

This module will raise a failure if the lower quartile for any base is less than 5 or if the median for any base is less than 20.

A.3.5.4. Common reasons for warnings

The most common reason for warnings and failures in this module is a general degradation of quality over the duration of long runs. In general sequencing chemistry degrades with increasing read length and for long runs you may find that the general quality of the run falls to a level where a warning or error is triggered.

A. *FastQC help*

If the quality of the library falls to a low level then the most common remedy is to perform quality trimming where reads are truncated based on their average quality. For most libraries where this type of degradation has occurred you will often be simultaneously running into the issue of adapter read-through so a combined adapter and quality trimming step is often employed.

Another possibility is that a warn / error is triggered because of a short loss of quality earlier in the run, which then recovers to produce later good quality sequence. This can happen if there is a transient problem with the run (bubbles passing through a flowcell for example). You can normally see this type of error by looking at the per-tile quality plot (if available for your platform). In these cases trimming is not advisable as it will remove later good sequence, but you might want to consider masking bases during subsequent mapping or assembly.

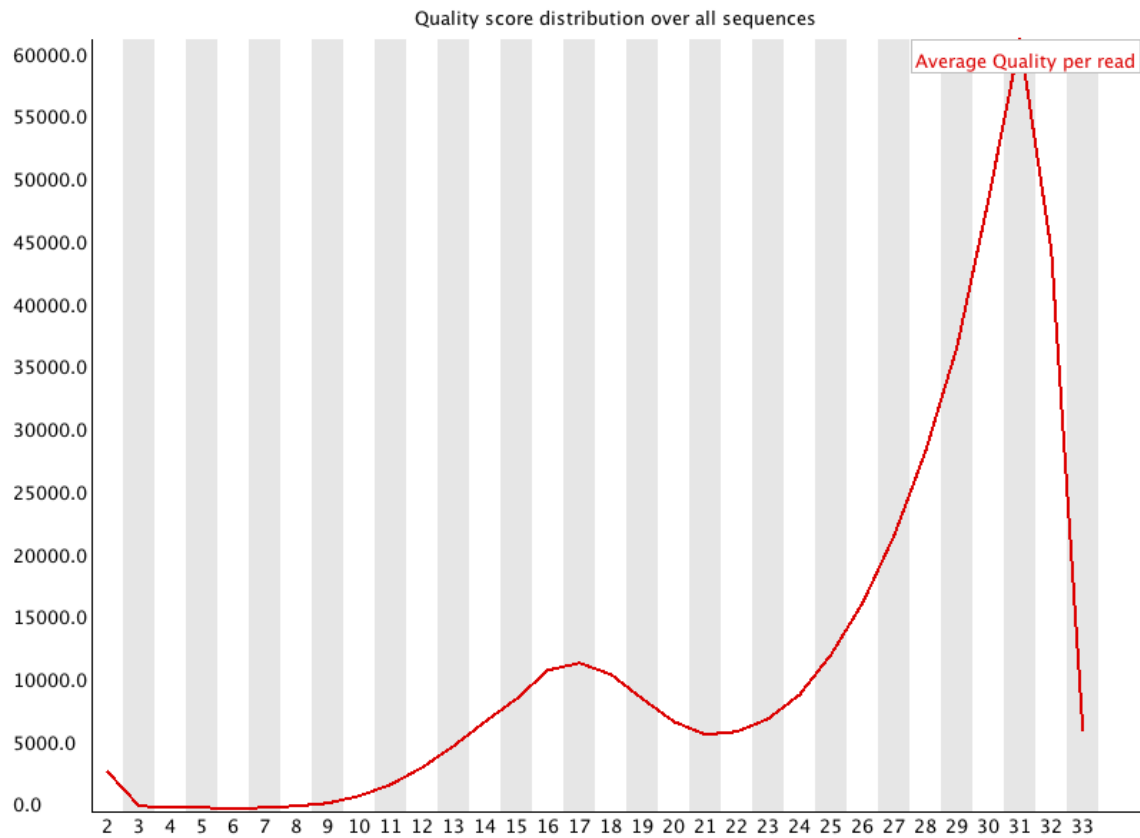
If your library has reads of varying length then you can find a warning or error is triggered from this module because of very low coverage for a given base range. Before committing to any action, check how many sequences were responsible for triggering an error by looking at the sequence length distribution module results.

A.3.6. Per Sequence Quality Scores

A.3.6.1. Summary

The per sequence quality score report allows you to see if a subset of your sequences have universally low quality values. It is often the case that a subset of sequences will have universally poor quality, often because they are poorly imaged (on the edge of the field of view etc), however these should represent only a small percentage of the total sequences.

A. FastQC help



If a significant proportion of the sequences in a run have overall low quality then this could indicate some kind of systematic problem - possibly with just part of the run (for example one end of a flowcell).

Results from this module will not be displayed if your input is a BAM/SAM file in which quality scores have not been recorded.

A.3.6.2. Warning

A warning is raised if the most frequently observed mean quality is below 27 - this equates to a 0.2% error rate.

A. *FastQC help*

A.3.6.3. Failure

An error is raised if the most frequently observed mean quality is below 20 - this equates to a 1% error rate.

A.3.6.4. Common reasons for warnings

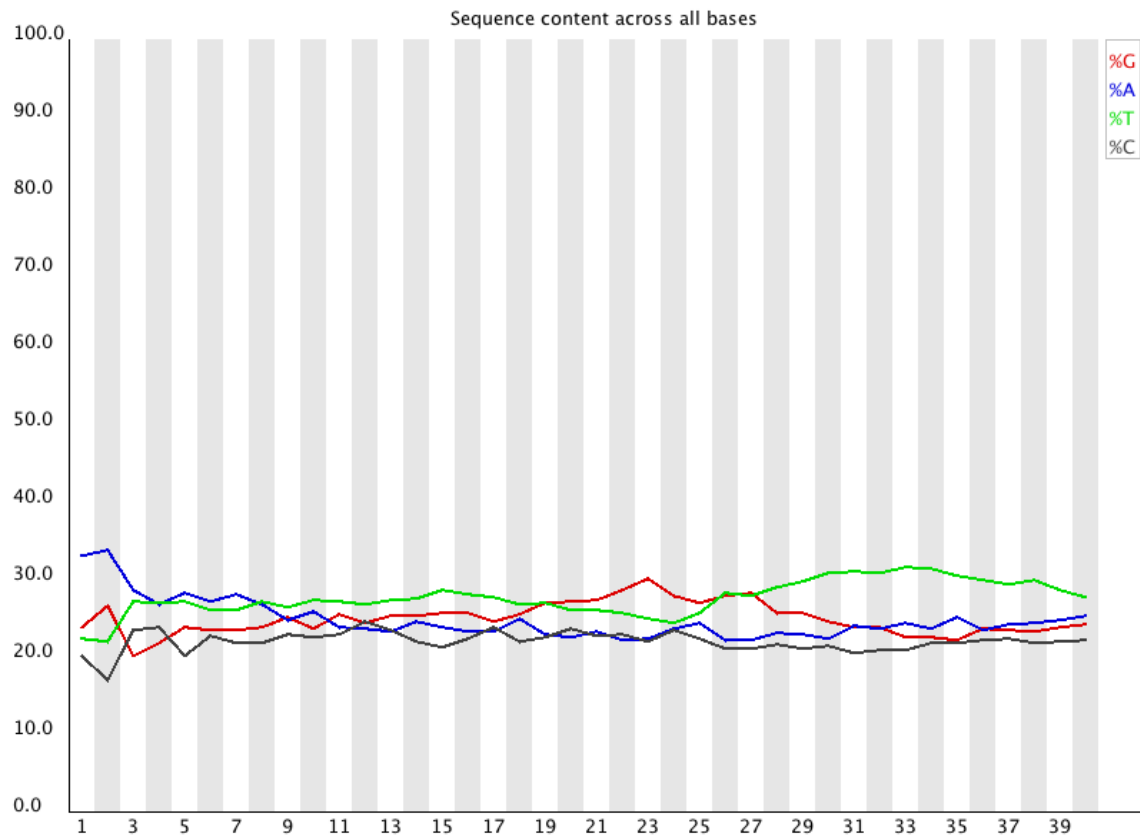
This module is generally fairly robust and errors here usually indicate a general loss of quality within a run. For long runs this may be alleviated through quality trimming. If a bi-modal, or complex distribution is seen then the results should be evaluated in concert with the per-tile qualities (if available) since this might indicate the reason for the loss in quality of a subset of sequences.

A.3.7. Per Base Sequence Content

A.3.7.1. Summary

Per Base Sequence Content plots out the proportion of each base position in a file for which each of the four normal DNA bases has been called.

A. FastQC help



In a random library you would expect that there would be little to no difference between the different bases of a sequence run, so the lines in this plot should run parallel with each other. The relative amount of each base should reflect the overall amount of these bases in your genome, but in any case they should not be hugely imbalanced from each other.

It's worth noting that some types of library will always produce biased sequence composition, normally at the start of the read. Libraries produced by priming using random hexamers (including nearly all RNA-Seq libraries) and those which were fragmented using transposases inherit an intrinsic bias in the positions at which reads start. This bias does not concern an absolute sequence, but instead provides enrichment of a number of different K-mers at the 5' end of the reads. Whilst this is a true technical bias, it isn't something which can be corrected by trimming and in most cases doesn't seem to adversely affect the downstream analysis. It will however produce a warning or error in this module.

A.3.7.2. Warning

This module issues a warning if the difference between A and T, or G and C is greater than 10% in any position.

A.3.7.3. Failure

This module will fail if the difference between A and T, or G and C is greater than 20% in any position.

A.3.7.4. Common reasons for warnings

There are a number of common scenarios which would elicit a warning or error from this module.

Overrepresented sequences: If there is any evidence of overrepresented sequences such as adapter dimers or rRNA in a sample then these sequences may bias the overall composition and their sequence will emerge from this plot.

Biased fragmentation: Any library which is generated based on the ligation of random hexamers or through tagmentation should theoretically have good diversity through the sequence, but experience has shown that these libraries always have a selection bias in around the first 12bp of each run. This is due to a biased selection of random primers, but doesn't represent any individually biased sequences. Nearly all RNA-Seq libraries will fail this module because of this bias, but this is not a problem which can be fixed by processing, and it doesn't seem to adversely affect the ability to measure expression.

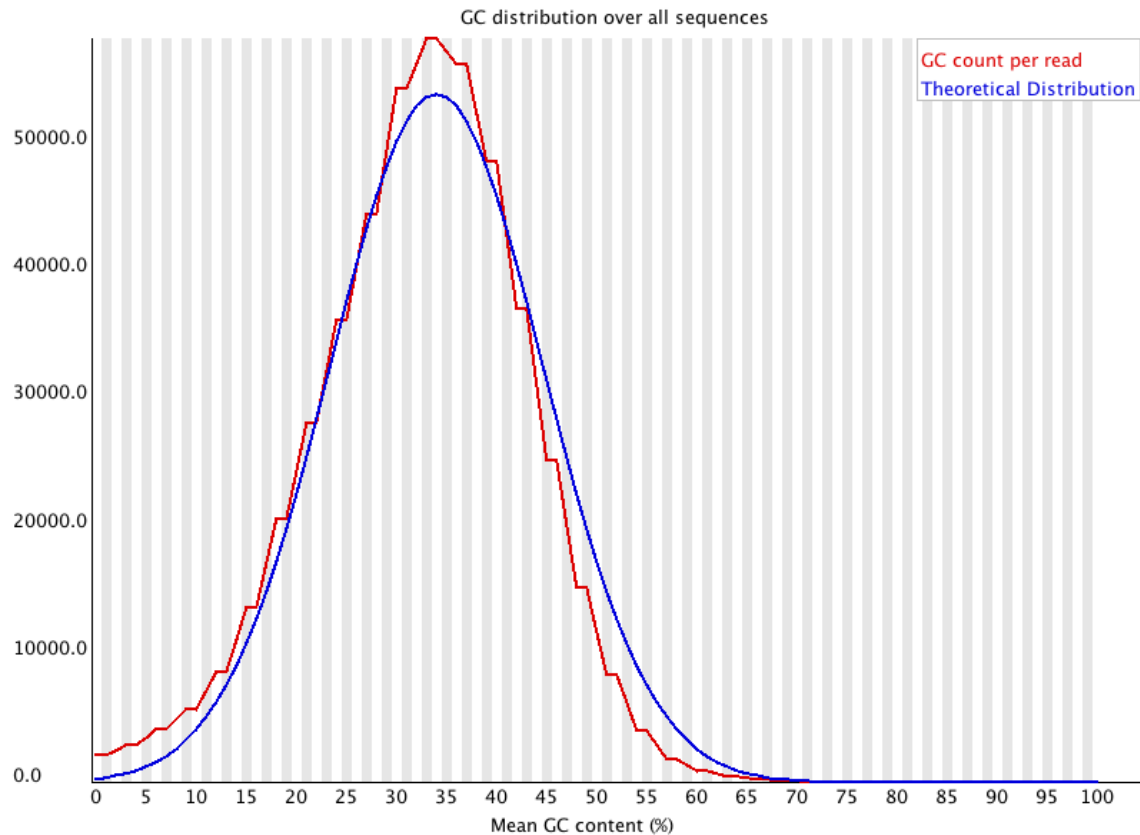
Biased composition libraries: Some libraries are inherently biased in their sequence composition. The most obvious example would be a library which has been treated with sodium bisulphite which will then have converted most of the cytosines to thymines, meaning that the base composition will be almost devoid of cytosines and will thus trigger an error, despite this being entirely normal for that type of library

If you are analysing a library which has been aggressively adapter trimmed then you will naturally introduce a composition bias at the end of the reads as sequences which happen to match short stretches of adapter are removed, leaving only sequences which do not match. Sudden deviations in composition at the end of libraries which have undergone aggressive trimming are therefore likely to be spurious.

A.3.8. Per Sequence GC Content

A.3.8.1. Summary

This module measures the GC content across the whole length of each sequence in a file and compares it to a modelled normal distribution of GC content.



In a normal random library you would expect to see a roughly normal distribution of GC content where the central peak corresponds to the overall GC content of the underlying genome. Since we don't know the the GC content of the genome the modal GC content is calculated from the observed data and used to build a reference distribution.

An unusually shaped distribution could indicate a contaminated library or some other kinds of biased subset. A normal distribution which is shifted indicates some systematic bias which is independent of base position. If there is a systematic bias which creates a shifted normal distribution then this won't be flagged as an error by the module since it doesn't know what your genome's GC content should be.

A. *FastQC help*

A.3.8.2. Warning

A warning is raised if the sum of the deviations from the normal distribution represents more than 15% of the reads.

A.3.8.3. Failure

This module will indicate a failure if the sum of the deviations from the normal distribution represents more than 30% of the reads.

A.3.8.4. Common reasons for warnings

Warnings in this module usually indicate a problem with the library. Sharp peaks on an otherwise smooth distribution are normally the result of a specific contaminant (adapter dimers for example), which may well be picked up by the overrepresented sequences module. Broader peaks may represent contamination with a different species.

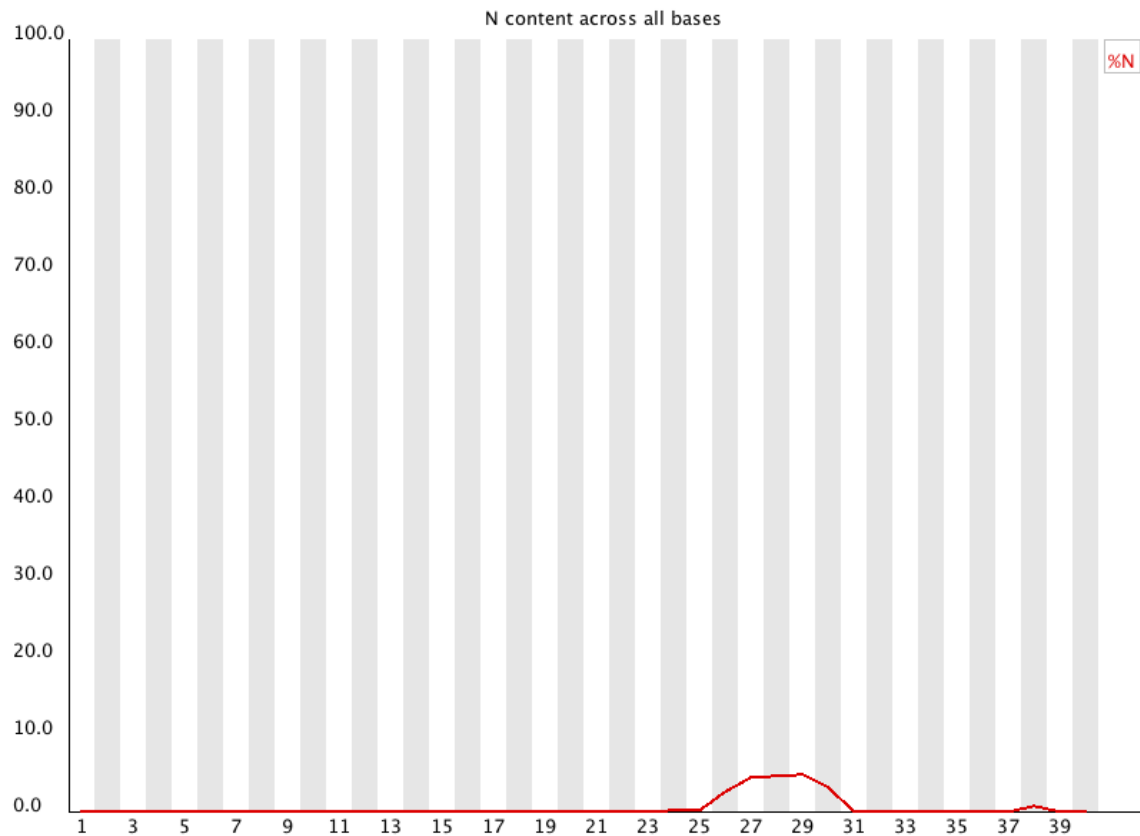
A.3.9. Per Base N Content

A.3.9.1. Summary

If a sequencer is unable to make a base call with sufficient confidence then it will normally substitute an N rather than a conventional base] call

This module plots out the percentage of base calls at each position for which an N was called.

A. FastQC help



It's not unusual to see a very low proportion of Ns appearing in a sequence, especially nearer the end of a sequence. However, if this proportion rises above a few percent it suggests that the analysis pipeline was unable to interpret the data well enough to make valid base calls.

A.3.9.2. Warning

This module raises a warning if any position shows an N content of >5%.

A.3.9.3. Failure

This module will raise an error if any position shows an N content of >20%.

A.3.9.4. Common reasons for warnings

The most common reason for the inclusion of significant proportions of Ns is a general loss of quality, so the results of this module should be evaluated in concert with those of the various quality modules. You should check the coverage of a specific bin, since it's possible that the last bin in this analysis could contain very few sequences, and an error could be prematurely triggered in this case.

Another common scenario is the incidence of a high proportions of N at a small number of positions early in the library, against a background of generally good quality. Such deviations can occur when you have very biased sequence composition in the library to the point that base callers can become confused and make poor calls. This type of problem will be apparent when looking at the per-base sequence content results.

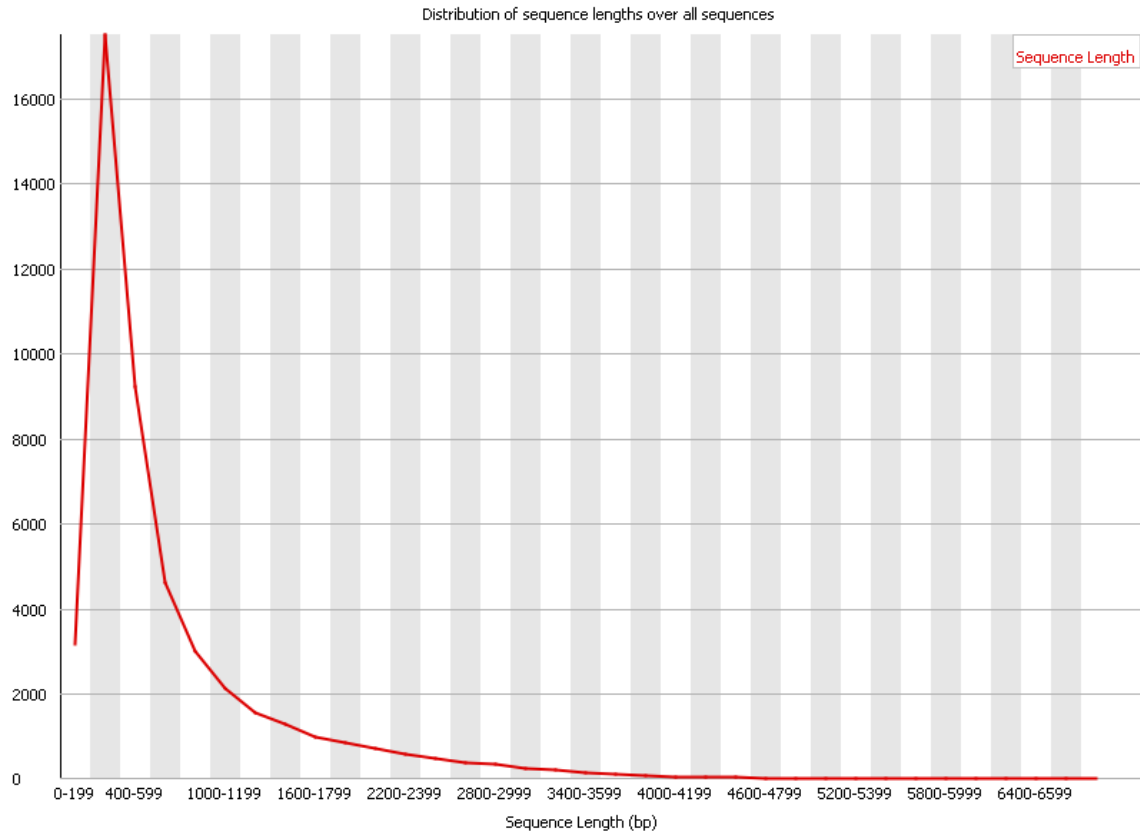
A.3.10. Sequence Length Distribution

A.3.10.1. Summary

Some high throughput sequencers generate sequence fragments of uniform length, but others can contain reads of wildly varying lengths. Even within uniform length libraries some pipelines will trim sequences to remove poor quality base calls from the end.

This module generates a graph showing the distribution of fragment sizes in the file which was analysed.

A. FastQC help



In many cases this will produce a simple graph showing a peak only at one size, but for variable length FastQ files this will show the relative amounts of each different size of sequence fragment.

A.3.10.2. Warning

This module will raise a warning if all sequences are not the same length.

A.3.10.3. Failure

This module will raise an error if any of the sequences have zero length.

A.3.10.4. Common reasons for warnings

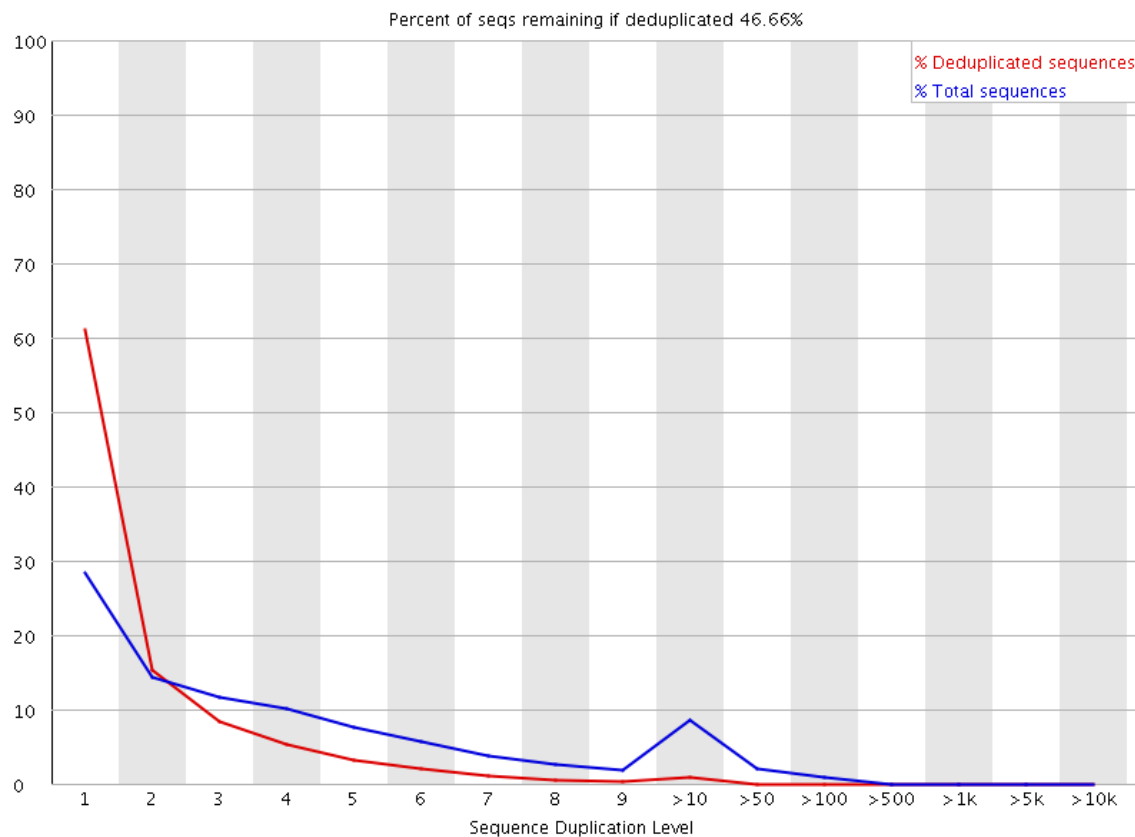
For some sequencing platforms it is entirely normal to have different read lengths so warnings here can be ignored.

A.3.11. Duplicate Sequences

A.3.11.1. Summary

In a diverse library most sequences will occur only once in the final set. A slightly elevated level of duplication may indicate a very high level of coverage of the target sequence, but a high level of duplication is more likely to indicate some kind of enrichment bias (eg PCR over amplification).

This module counts the degree of duplication for every sequence in a library and creates a plot showing the relative number of sequences with different degrees of duplication.



A. *FastQC help*

To cut down on the memory requirements for this module only sequences which first appear in the first 100,000 sequences in each file are analysed, and these are then tracked through the entire dataset. This should be enough to get a good impression for the duplication levels in the whole file. Each sequence is tracked to the end of the file to give a representative count of the overall duplication level. To cut down on the amount of information in the final plot any sequences with more than 10 duplicates are placed into grouped bins to give a clear impression of the overall duplication level without having to show each individual duplication value.

Because the duplication detection requires an exact sequence match over the whole length of the sequence, any reads over 50bp in length are truncated to 50bp for the purposes of this analysis. Even so, longer reads are more likely to contain sequencing errors which will artificially increase the observed diversity and will tend to underrepresent highly duplicated sequences.

The plot shows the proportion of the library which is made up of sequences in each of the different duplication level bins. The x axis is the degree of duplication, and the y axis shows the proportion of the library which is made up of reads with that level of duplication.

In a properly diverse library most sequences should fall into the far left of the plot . A general level of enrichment, indicating broad oversequencing in the library will tend to flatten the line, lowering the low end and generally raising other categories. More specific enrichments of subsets, or the presence of low complexity contaminants will tend to produce spikes towards the right of the plot.

The module also calculates an expected overall loss of sequence were the library to be deduplicated. This headline figure is shown at the top of the plot and gives a reasonable impression of the potential overall level of loss.

A.3.11.2. Warning

This module will issue a warning if non-unique sequences make up more than 30% of the total.

A.3.11.3. Failure

This module will issue a error if non-unique sequences make up more than 50% of the total.

A.3.11.4. Common reasons for warnings

The underlying assumption of this module is of a diverse unenriched library. Any deviation from this assumption will naturally generate duplicates and can lead to warnings or errors from this module.

In general there are two potential types of duplicate in a library, technical duplicates arising from PCR artefacts, or biological duplicates which are natural collisions where different copies of exactly the same sequence are randomly selected. From a sequence level there is no way to distinguish between these two types and both will be reported as duplicates here.

A warning or error in this module is simply a statement that you have exhausted the diversity in at least part of your library and are re-sequencing the same sequences. In a supposedly diverse library this would suggest that the diversity has been partially or completely exhausted and that you are therefore wasting sequencing capacity. However in some library types you will naturally tend to over-sequence parts of the library and therefore generate duplication and will therefore expect to see warnings or error from this module.

In RNA-Seq libraries sequences from different transcripts will be present at wildly different levels in the starting population. In order to be able to observe lowly expressed transcripts it is therefore common to greatly over-sequence high expressed transcripts, and this will potentially create large set of duplicates. This will result in high overall duplication in this test, and will often produce peaks in the higher duplication bins. This duplication will come from physically connected regions, and an examination of the distribution of duplicates in a specific genomic region will allow the distinction between over-sequencing and general technical duplication, but these distinctions are not possible from raw fastq files. A similar situation can arise in highly enriched ChIP-Seq libraries although the duplication there is less pronounced. Finally, if you have a library where the sequence start points are constrained (a library constructed around restriction sites for example, or an unfragmented small RNA library) then the constrained start sites will generate huge duplication levels which should not be treated as a problem, nor removed by deduplication. In these types of library you should consider using a system such as random barcoding to allow the distinction of technical and biological duplicates.

A.3.12. Overrepresented Sequences

A.3.12.1. Summary

A normal high-throughput library will contain a diverse set of sequences, with each individual sequence making up only a tiny fraction of the whole. Finding that a single sequence is very overrepresented in the set either means that it is highly biologically significant, or indicates that the library is contaminated, or not as diverse as you expected.

This module lists all of the sequences which make up more than 0.1% of the total. To conserve memory only sequences which appear in the first 100,000 sequences are tracked to the end of the file. It is therefore possible that a sequence which is overrepresented but doesn't appear at the start of the file for some reason could be missed by this module.

For each overrepresented sequence the program will look for matches in a database of common contaminants and will report the best hit it finds. Hits must be at least 20bp in length and have no more than 1 mismatch. Finding a hit doesn't necessarily mean that this is the source of the contamination, but may point you in the right direction. It's also worth pointing out that many adapter sequences are very similar to each other so you may get a hit reported which isn't technically correct, but which has very similar sequence to the actual match.

Because the duplication detection requires an exact sequence match over the whole length of the sequence any reads over 50bp in length are truncated to 50bp for the purposes of this analysis. Even so, longer reads are more likely to contain sequencing errors which will artificially increase the observed diversity and will tend to underrepresent highly duplicated sequences.

A.3.12.2. Warning

This module will issue a warning if any sequence is found to represent more than 0.1% of the total.

A.3.12.3. Failure

This module will issue an error if any sequence is found to represent more than 1% of the total.

A. FastQC help

A.3.12.4. Common reasons for warnings

This module will often be triggered when used to analyse small RNA libraries where sequences are not subjected to random fragmentation, and the same sequence may naturally be present in a significant proportion of the library.



Universidad de **Nariño**
FUNDADA EN 1904



Universidad de **Nariño**
ACREDITADA DE ALTA CALIDAD
RESOLUCIÓN MEN 10567 - MAYO 23 DE 2017



C E S U N
CENTRO DE ESTUDIOS EN SALUD
Universidad de **Nariño**