Bitacora: calidad de la secuenciación masiva y ensamblaje

DNA obtenido de la tesis de Diamanda Tapia Gallardo (https://biblioteca.cicese.mx/catalogo/tesis/ficha.php?id=25475)

In [ ]:
```
EDITH ELIZONDO
```

En esta bitácora se utilizarán diferentes programas

FastQC

Se puede encontrar más información en el sitio de fastqc (http://www.bioinformatics.babraham.ac.uk/projects/fastqc/)

y de Trim Galore (https://www.bioinformatics.babraham.ac.uk/projects/trim_galore/)

Ensamblaje con Soapdenovo2

Luo, R. et al. 2012. SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler. Gigascience 1, 18. doi: 10.1186/2047-217X-1-18 (https://academic.oup.com/gigascience/article/1/1/2047-217X-1-18/2656146) SOAPdenovo2 manual (https://github.com/aquaskyline/SOAPdenovo2)

Todos estos programas se pueden instalar en sistemas Linux o iOS desde anaconda (https://anaconda.org/bioconda/soapdenovo2) (altamente recomendable)

In [ ]:
```
import os
from Bio import SeqIO, Entrez
```

La función `cuentasecuencias` cuenta el número de secuencias encontradas en el archivo `*.gz` sin descomprimir el archivo.

In [ ]:
```
def cuentasecuencias(arch1):
    count, total_len, maximo = 0, 0, 0,
    minimo = 1000
    with gzip.open(arch1, "rt") as handle:
        for rec in SeqIO.parse(handle, "fastq"):
            count += 1
            total_len += len(rec.seq)
            if maximo< len(rec.seq):
                maximo = len(rec.seq)
            if minimo > len(rec.seq):
                minimo = len(rec.seq)
    return (count, total_len, maximo, minimo)
```

In [ ]:
```
cd /LUSTRE/bioinformatica_data/lga/edith/data/microalgas/
```

In [ ]:
```
ls
```

In [ ]:
```
ls /LUSTRE/bioinformatica_data/lga/edith/data/microalgas/
```

In [ ]:
```
!head -13 numero_lecturas.csv
```

Fastqc

Se procesan los archivos en lotes para obtener el análisis de calidad de los archivos. En este caso se habían descomprimido los archivos, pero se pueden utilizar los archivos comprimidos *.gz, los resultados de salida estarán comprimidos también

verificar fastqc

In [ ]:
```
!fastqc -help
```

In [ ]:
```
ls
```

In [ ]:
```
%%bash
for file in *.fastq
do
 echo "procesando", $file
 time fastqc $file
done
```

In [ ]:
```
ls -lh *.html
```

In [ ]:
```
ls -lh *.fastq
```

Trim_galore

Condiciones de recorte con trim_galore

-q se eliminarán los extremos de baja calidad, además de eliminar los adaptadores.
--fastqc al finalizar se ejecutará nuevamente el `fastqc` --paired se tienen lecturas pareadas --retain_unpaired se retienen lecturas que tengan diferentes tamaños con su lectura pareada

NOTA: en este caso debido a que se requieren dos archivos por corrida, se desplegaron los archivos, pero se puede realizar un comando que requiera un ciclo (loop)

In [ ]:
```
%%bash
time trim_galore --fastqc --retain_unpaired \
 --paired CA2_S27_L001_R1_001.fastq CA2_S27_L001_R2_001.fastq
```

In [ ]:
```
%%bash
time trim_galore --fastqc --retain_unpaired \
 --paired CA2_S27_L001_R1_001.fastq  CA2_S27_L001_R2_001.fastq

time trim_galore --fastqc --retain_unpaired \
 --paired CN2_S28_L001_R1_001.fastq  CN2_S28_L001_R2_001.fastq

time trim_galore --fastqc --retain_unpaired \
 --paired MA2_S25_L001_R1_001.fastq  MA2_S25_L001_R2_001.fastq

time trim_galore --fastqc --retain_unpaired \
 --paired MN2_S26_L001_R1_001.fastq  MN2_S26_L001_R2_001.fastq

time trim_galore --fastqc --retain_unpaired \
 --paired XA2_S29_L001_R1_001.fastq  XA2_S29_L001_R2_001.fastq

time trim_galore --fastqc --retain_unpaired \
 --paired XN2_S30_L001_R1_001.fastq  XN2_S30_L001_R2_001.fastq
```

In [ ]:
```
ls -lh *.html
```

Visualización de las graficas

In [ ]:
```
ls *.zip
```

In [ ]:
```
from zipfile import ZipFile
import os
```

In [ ]:
```
os.makedirs('img',exist_ok=True)
```

In [ ]:
```
cd img
```

In [ ]:
```
lista = !ls ../*fastqc.zip
lista.sort()
lista
```

Extrae archivos `png` y los separa por directorios

In [ ]:
```
for row in lista:
    with ZipFile(row, 'r') as zipObj:
       # Get a list of all archived file names from the zip
       listOfFileNames = zipObj.namelist()
       # Iterate over the file names
       for fileName in listOfFileNames:
           # Check filename endswith csv
           if fileName.endswith('.png') and fileName.find("Images")!=-
1:
                # Extract a single file from zip
                zipObj.extract(fileName)
                print("extrae ", fileName)
```

In [ ]:
```
lista1 = !ls
lista1
```

In [ ]:
```
from IPython.display import display, Image
```

Edith

In [ ]:
```
for row1 in lista1:  #directorio
    if row1[:1]!=".":
        directorio = row1 + "/"
        #print(directorio)
        lista2 = os.listdir(directorio)
        for row2 in lista2:    #subdirectorio
            if row2[:1]!=".":
                #print(directorio+row2,"\t", row2)
                lista3 = os.listdir(directorio+row2)   #display(Image(
directorio+row2, width=200))
                for row3 in lista3: #imagenes

                    print(directorio+row2+"/"+ row3)
                    display(Image(directorio+row2+"/"+row3, width=200)
)
```

In [ ]:
```
for row1 in lista1:  #directorio
    if row1[:1]!=".":
        directorio = row1 + "/"
        #print(directorio)
        lista2 = os.listdir(directorio)
        for row2 in lista2:    #subdirectorio
            if row2[:1]!=".":
                #print(directorio+row2,"\t", row2)
                lista3 = os.listdir(directorio+row2)   #display(Image(
directorio+row2, width=200))
                for row3 in lista3: #imagenes

                    print(directorio+row2+"/"+ row3)
                    display(Image(directorio+row2+"/"+row3, width=200)
)
```

In [ ]:
```
pwd
```

Para observar solo las gráficas de calidad

In [ ]:
```
for row1 in lista1:  #directorio
    if row1[:1]!=".":
        directorio = row1 + "/"
        #print(directorio)
        lista2 = os.listdir(directorio)
        for row2 in lista2:    #subdirectorio
            if row2[:1]!=".":
                #print(directorio+row2,"\t", row2)
                lista3 = os.listdir(directorio+row2)   #display(Image(
directorio+row2, width=200))
                for row3 in lista3: #imagenes
                    if row3=="per_base_quality.png":
                        print(directorio+row2+"/"+ row3)
                        display(Image(directorio+row2+"/"+row3, width=
200))
```

Para observar solo las gráficas de la calidad de las celdas de flujo

In [ ]:
```
for row1 in lista1:  #directorio
    if row1[:1]!=".":
        directorio = row1 + "/"
        #print(directorio)
        lista2 = os.listdir(directorio)
        for row2 in lista2:    #subdirectorio
            if row2[:1]!=".":
                #print(directorio+row2,"\t", row2)
                lista3 = os.listdir(directorio+row2)   #display(Image(
directorio+row2, width=200))
                for row3 in lista3: #imagenes
                    if row3=="per_tile_quality.png":
                        print(directorio+row2+"/"+ row3)
                        display(Image(directorio+row2+"/"+row3, width=
200))
```

Ejercicio:

¿Qué significan los colores rojos, amarillos y azul-verde observados en los esquemas de la celda anterior

In [ ]:
```
def cuentasecuencias(arch1):
    count, total_len, maximo = 0, 0, 0,
    minimo = 100
    for rec in SeqIO.parse(arch1, "fastq"):
        count += 1
        total_len += len(rec.seq)
        if maximo< len(rec.seq):
            maximo = len(rec.seq)
        if minimo > len(rec.seq):
            minimo = len(rec.seq)
    return (count, total_len, maximo, minimo)
```

In [ ]:
```
cd ..
```

In [ ]:
```
ls *.fq
```

In [ ]:
```
%%bash
for file in *.fq.gz
do
 echo "procesando", $file
 time zgrep "^@" $file | wc -l
done
```

In [ ]:
```
%%bash
time for file in *.fq.gz
do
 echo $file >> lecturas_fq_gz.txt
 zgrep "^@" $file | wc -l >> lecturas_fq_gz.txt
done
```

In [ ]:
```
cd /LUSTRE/bioinformatica_data/lga/edith/data/microalgas/
```

In [ ]:
```
f = open("lecturas.txt", "r")
lineas =[]
n=1
lin =""
print( "\tarchivo\t\t\tnúmero de secuencias")
for row in f:
    lin = row
    lin = lin.replace("\n", "")
    lin = lin.replace(" ","")

    if n%2!=0:
        print (n, lin, end = "")
        lin1 = lin
    else:
        #lin = row
        print("\t", lin)
        lin1 = lin1,lin
        lineas.append(lin1)
        lin1=[]
    n+=1
```

se suspendio la ejecucion de la bitacora y se esta retomando a partir de este punto.

se vuelven a cargar los paquetes

In [ ]:
```
from pandas import DataFrame
import pandas as pd
```

In [ ]:
```
f1 = DataFrame(lineas, index=None, columns =("archivo", "numero_de_lec
turas"))
f1.to_csv("numero_lecturas_fq_gz.csv", index= None)
f1
```

In [ ]:
```
pwd
```

In [ ]:
```
ls *.csv
```

In [ ]:
```
f1 =pd.read_csv("numero_lecturas.csv")
f1.head()
```

In [ ]:
```
serie = f1["archivo"].str[:7]
f1["arc"] = serie
f1
```

In [ ]:
```
#seborralosultimostresdatos
f1.drop([36,37,38], inplace=True)
f1
```

In [ ]:
```
f1.to_csv("numero_lecturas_arc.csv", index=None)
```

In [ ]:
```
ls
```

In [ ]:
```
import gzip
from Bio import SeqIO
```

In [ ]:
```
pwd
```

In [ ]:
```
lista = !ls *.fq
lista.sort()
lista
```

explicar comando

In [ ]:
```
print("{:<36s} {:>8} {:>5} {:>5}".format("archivo", "#lecturas", "min"
, "max"))
secuencias=[]
for row in lista:
    count1, total_len1, maximo1, minimo1 = cuentasecuencias(row)
    print("{:<36s} {:>8} {:>5} {:>5}".format(row, count1, minimo1, max
imo1))
    secuencias.append((row, count1, minimo1, maximo1))
```

In [ ]:
```
secuencias = DataFrame(secuencias, index=None, columns =("archivo", "l
ecturas" , "minimo" , "maximo"))
secuencias.to_csv("numero_lecturas_fq.csv", index= None)
secuencias
```

In [ ]:
```
pwd
```

In [ ]:
```
cd /LUSTRE/bioinformatica_data/lga/edith/data/microalgas/
```

In [ ]:
```
lista = !ls *.fastq
lista.sort()
lista
```

In [ ]:
```
print("{:<38s} {:>10} {:>5} {:>5}".format("archivo", "#lecturas", "min
", "max"))
secuencias = []
for row in lista:
    count1, total_len1, maximo1, minimo1 = cuentasecuencias(row)
    print("{:<38s} {:>10} {:>5} {:>5}".format(row[row.find("/")+1:], c
ount1, minimo1, maximo1))
    secuencias.append((row[row.find("/")+1:], count1, minimo1, maximo1
))
```

In [ ]:
```
secuencias = DataFrame(secuencias, index=None, columns =("archivo", "l
ecturas" , "minimo" , "maximo"))
secuencias.to_csv("numero_lecturas_arc.csv", index = None)
secuencias
```

In [ ]:
```
sec = pd.read_csv("numero_lecturas_arc.csv",  sep = ",", index_col =0)
sec1 = pd.read_csv("numero_lecturas_fq.csv",  sep = ",", index_col =0)
print(sec.head())
print(sec1.head())
```

In [ ]:
```
sec2 = sec.append(sec1, sort = True)
sec2.sort_values(by= "archivo", inplace = True)
sec2
```

In [ ]:
```
sec2.reset_index(inplace= True)
sec2
```

In [ ]:
```
sec2 = sec.append(sec1, sort = True)
sec2.sort_values(by= "archivo", inplace = True)
sec2
```

In [ ]:
```
sec2.to_csv("numero_lecturas.csv", index = None)
```

Archivo de configuración para el ensamblador Soapdenovo

En caso de ser necesario, cambiar el valor de `max_rd_len=`, en este caso a **301**
Revisar los demás parámetros, p. e. `rd_len_cutoff=`
Cambie las líneas q1 y q2, en este caso a:
`q1= nombre_del_archivo_R1`
`q2= nombre_del_archivo_R2`

Leer el manual (https://vcru.wisc.edu/simonlab/bioinformatics/programs/soap/SOAPdenovo2MANUAL.txt).
Asimismo es necesario tomar en cuenta el nombre del archivo de la configuración. En este caso:
`nombre_del_archivo_soap.txt`

In [ ]:
```
lista = !ls *.fq
lista.sort()
#lista
print("{:<36s}".format("archivo"))
for row in lista:
    if row.find("unpaired")==-1:
        print("{:<36s} ".format(row))
```

In [ ]:
```
os.makedirs('soap_config',exist_ok=True)
```

In [ ]:
```
def soap(arc0,arc):
    arc1 = "soap_config/" +arc0[:arc0.find("_")] +".txt"
    #arc1 = "soap_config/" + arc1
    print("archivo de configuracion SOAPDENOVO=", arc1)

    fout = open(arc1, "w")
    linea="""#maximal read length
max_rd_len=301
[LIB]
#average insert size of the library
avg_ins=301
#if sequences are forward-reverse of reverse-forward
reverse_seq=0
#in which part(s) the reads are used (only contigs, only scaffolds, bo
th contigs and scaffolds, only gap closure)
asm_flags=3
#cut the reads to the given length
rd_len_cutoff=300
#in which order the reads are used while scaffolding
rank=1
# cutoff of pair number for a reliable connection (at least 3 for shor
t insert size)
pair_num_cutoff=3
#minimum aligned length to contigs for a reliable read location (at le
ast 32 for short insert size)
map_len=32
#paired-end fastq files, read 1 file should always be followed by read
2 file
"""

    linea +="q1=" + arc0 +"\n"
    linea += "q2="+ arc +"\n"
    linea+="""#another pair of paired-end fastq files, read 1 file sho
uld always be followed by read 2 file
#q1=input_reads2_pair_1.fq
#q2=input_reads2_pair_2.fq
#paired-end fasta files, read 1 file should always be followed by read
2 file
#f1= SRX5014491_1_val_1.fq
#f2=SRX5014491_2_val_2.fq
#fastq file for single reads
#q=input_reads.fq
"""

    fout.write(linea)

    fout.close()
    return (linea)
```

In [ ]:
```
def sh_ejecutar(arc0,directo):
    arc1 = "soap_config/" +arc0[:arc0.find("_")] +".txt"
    arc2 = "soap_config/" +arc0[:arc0.find("_")] +".sh"
    fout = open(arc2, "w")
    print("archivo de salida de soapdenovo", arc2)
    linea="""#!/bin/sh
#SBATCH --job-name=SOAPdenovo2
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=8
#SBATCH --time=168:00:00
#SBATCH --mem=50gb
#SBATCH --output=SOAPdenovo2.%J.out
#SBATCH --error=SOAPdenovo2.%J.err

#module load soapdenovo2/r240

SOAPdenovo-63mer all -s """
    linea += arc1
    linea +=" -K 31 -o "
    linea += directorio+ "/" + directorio+"_out "
    linea +="""-p $SLURM_NTASKS_PER_NODE


"""

    fout.write(linea)

    fout.close()
    return (linea)
```

In [ ]:
```
n=0
print("{:<36s} {:<36s}".format("archivo1", "archivo2"))
for row in lista:
    if row.find("unpaired")==-1:

        if n==1:
            print("{:<36s} ".format(row))
            n=0
            archivo1=row
            soap(archivo0, archivo1)
            directorio = archivo0[:archivo0.find("_")]
            os.makedirs(directorio,exist_ok=True)
            print("creando directorio", directorio)
            sh_ejecutar(archivo0, directorio)
        else:
            print("{:<36s} ".format(row), end ="")
            archivo0=row
```

In [ ]:
```
n=0
print("{:<36s} {:<36s}".format("archivo1", "archivo2"))
for row in lista:
    if row.find("unpaired")==-1:

        if n==1:
            print("{:<36s} ".format(row))
            n=0
            archivo1=row
            soap(archivo0, archivo1)
            directorio = archivo0[:archivo0.find("_")]
            os.makedirs(directorio,exist_ok=True)
            print("creando directorio", directorio)
            sh_ejecutar(archivo0, directorio)
        else:
            print("{:<36s} ".format(row), end ="")
            archivo0=row
            n+=1
```

In [ ]:
```
ls soap_config/
```

Verificando contenido del archivo de configuración

In [ ]:
```
!head -50 soap_config/CA2.txt
```

Se verifica el contenido del archivo de ejecución con sh

In [ ]:
```
!head -20 soap_config/CN2.sh
```

Se ejecuta el programa

In [ ]:
```
%%bash
for file in soap_config/*.sh
do
 echo $file
 time sh $file
done
```

Se verifican los archivos de salida y su contenido

In [ ]:
```
from pandas import Series
```

In [ ]:
```
lista = !ls *.fq
lista.sort()
listado =[]
duplicado = ""
for row in lista:
    if row[:3]!=duplicado:
        listado.append(row[:3])
        print(row[:3])
        duplicado=row[:3]
```

In [ ]:
```
for row in listado:
    n, n1 = 0, 0
    directorio = row +"/"
    archivo = directorio + row+ "_out.contig"
    archivo_fasta = directorio + row+ ".fasta"
    secuencias=[]
    print(archivo,  end = "\t")
    for rec in SeqIO.parse(archivo, "fasta"):
        n+=1
        if len(rec.seq)>200:
            #if n1%500==0:
            #    print ("{:>9} {:>8} cobertura {:>5} {:>5}".format(n1,
rec.id, rec.description[rec.description.find("_")+1:][:rec.description
[rec.description.find("_")+1:].find("_")],
            #           len(rec.seq)))
            secuencias.append(rec)
            n1 +=1
    SeqIO.write(secuencias, archivo_fasta, 'fasta')
    print("secuencias totales {:>7} secuencias mayores de 200 pb {:>6}
".format( n, n1))
```

In [ ]:
```
cd /LUSTRE/bioinformatica_data/lga/edith/data/microalgas/
```

Los archivos con los ensamblajes se encuentran dentro de cada directorio de las cepas y están en formato fasta, pero tienen la terminación `_out.contig`

In [ ]:
```
ls CA2/
```

In [ ]:
```
!head CA2/CA2_out.contig
```

Estadística

In [ ]:
```
!head -100 CA2/CA2_out.scafStatistics
```

In [ ]:
```
!head -100 CA2/CA2_out.kmerFreq
```

Una vez concluido el ensamblaje se procede a la anotación de los contigs