Anexo

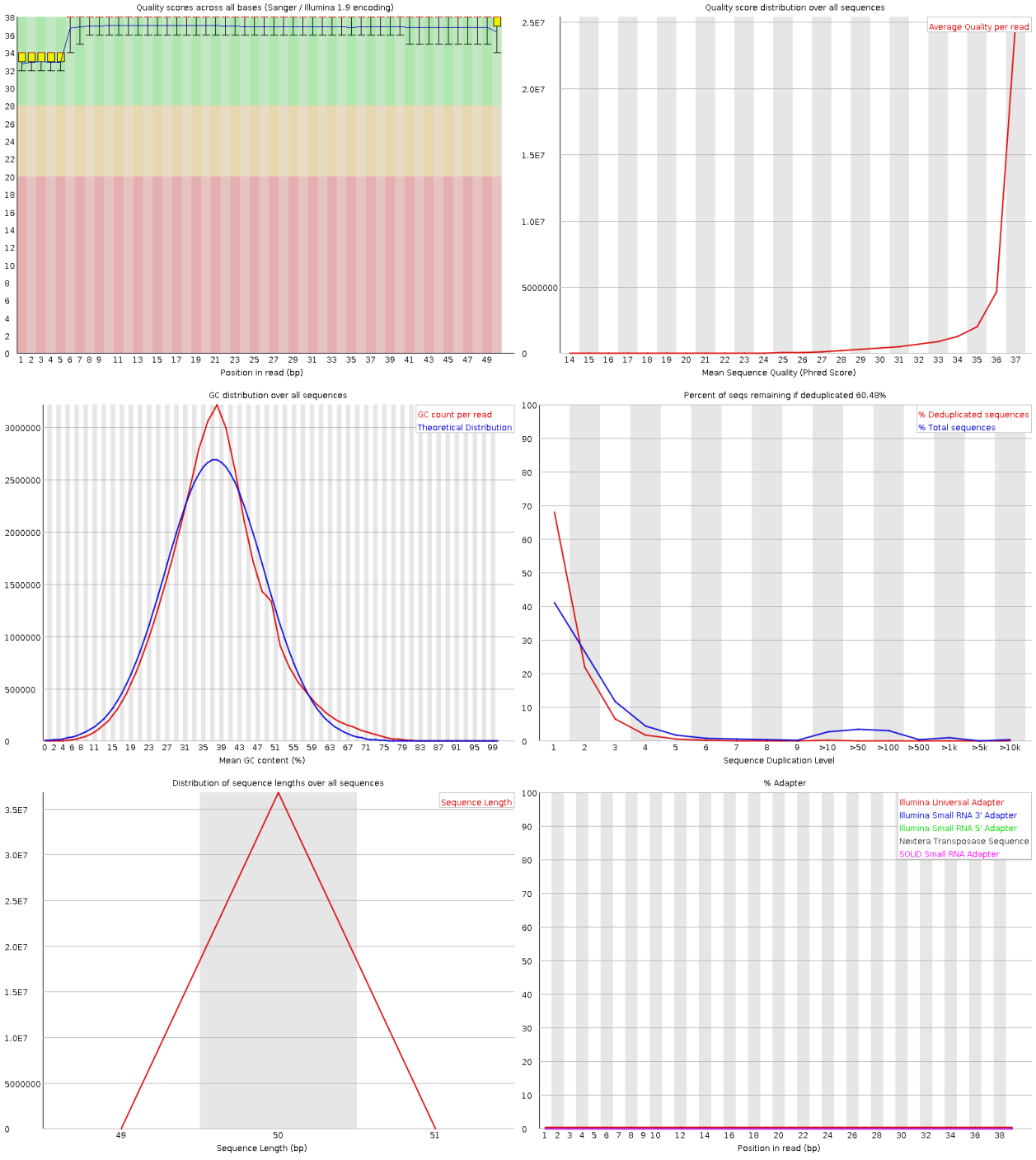

Figura 5.1. Análisis de calidad de las lecturas del control empleando fastQC. Muestra la calidad por base, calidad por secuencia, contenido en GC por secuencia, niveles de duplicación, distribución de la longitud de las lecturas y contenido de adaptadores.

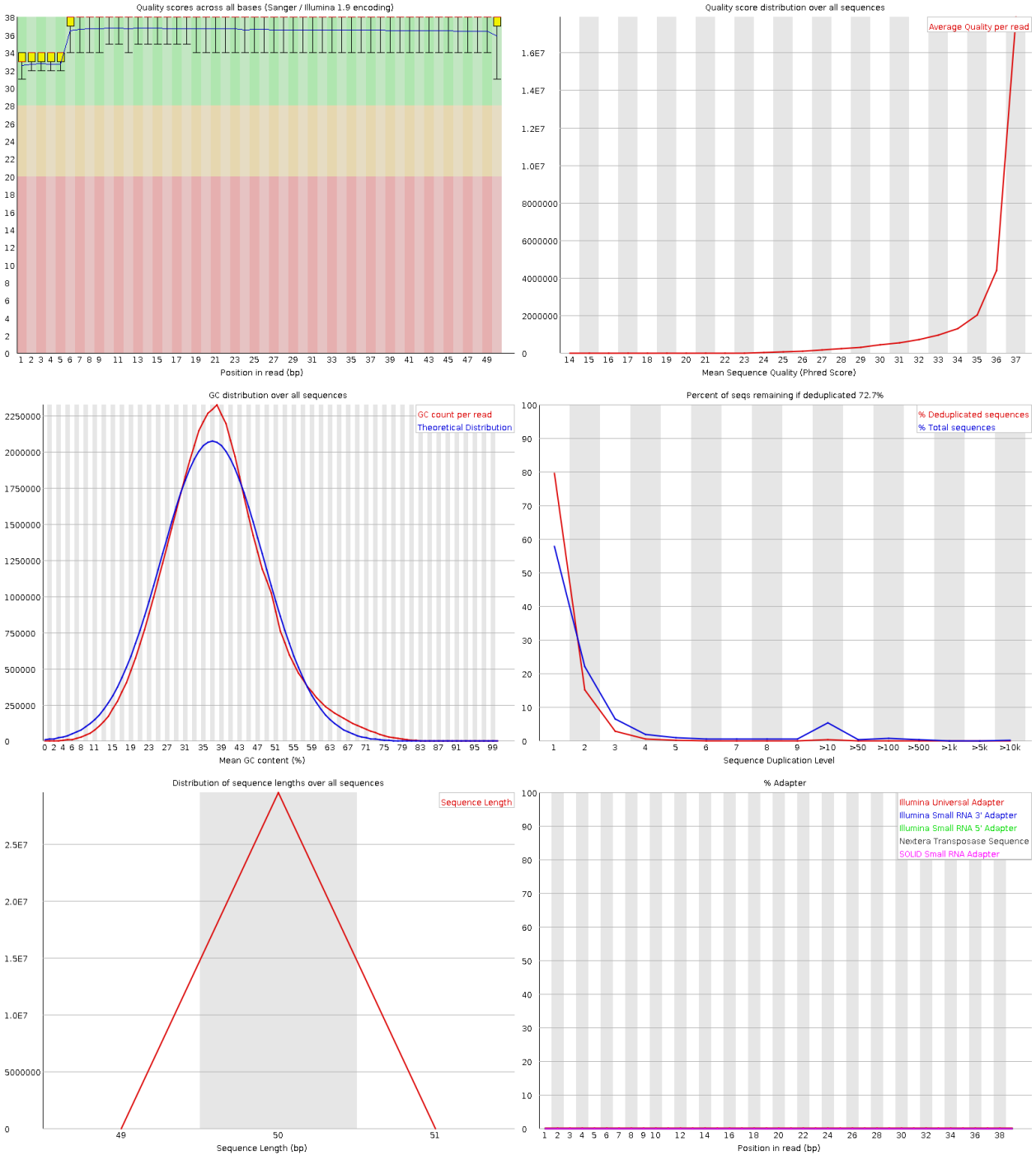

Figura 5.2. Análisis de calidad de las lecturas de la réplica 1 empleando fastQC. Muestra la calidad por base, calidad por secuencia, contenido en GC por secuencia, niveles de duplicación, distribución de la longitud de las lecturas y contenido de adaptadores.

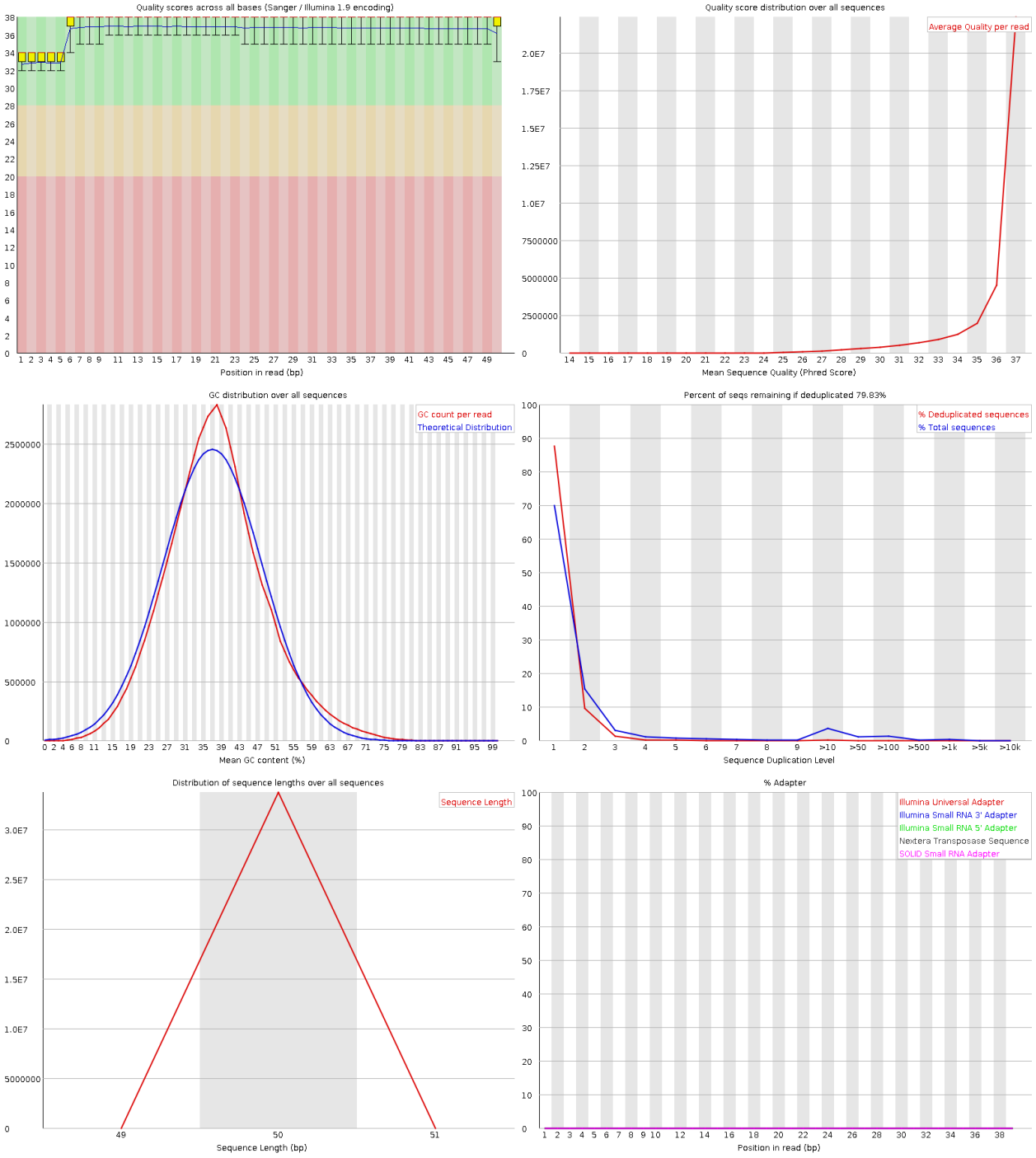

Figura 5.3. Análisis de calidad de las lecturas de la réplica 2 empleando fastQC. Muestra la calidad por base, calidad por secuencia, contenido en GC por secuencia, niveles de duplicación, distribución de la longitud de las lecturas y contenido de adaptadores.

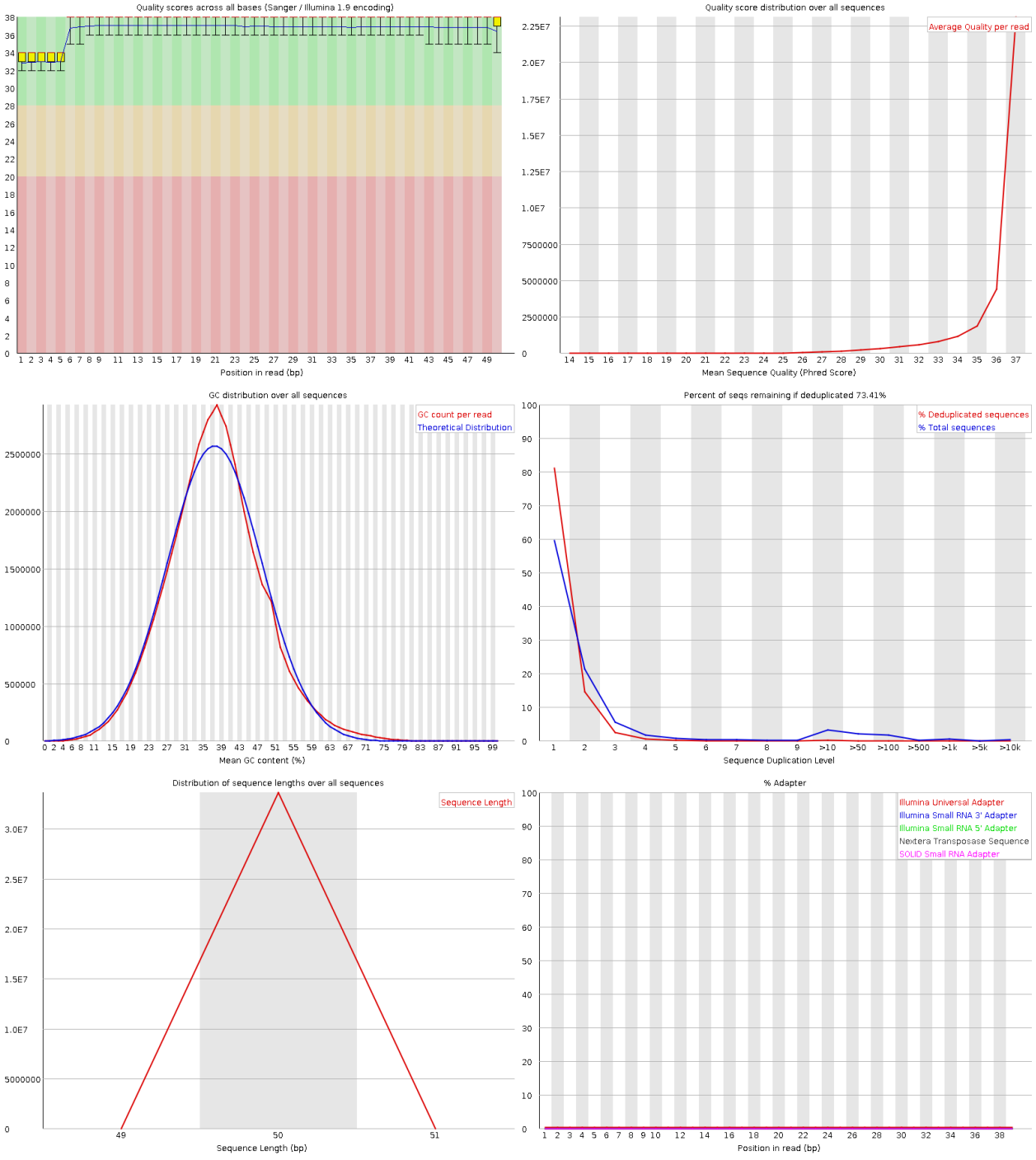

Figura 5.4. Análisis de calidad de las lecturas de la réplica 3 empleando fastQC. Muestra la calidad por base, calidad por secuencia, contenido en GC por secuencia, niveles de duplicación, distribución de la longitud de las lecturas y contenido de adaptadores.

```bash
#!/bin/bash

# Este script sirve para procesar lecturas de ChIP-seq
# desde el control de calidad hasta la llamada de picos

# Parametros
fastq_dir="fastq"
data_tsv="data.tsv"
control="ler_rep1"

genome_fasta="TAIR10.fa"
genome_size=119481543
threads=4

# 1. FastQC
mkdir -p results/fastqc logs/fastqc/

cat $data | tail -n +2 | while read name group rep fastq; do
  fastqc \
    -o results/fastqc/ \
    $fastq_dir/$fastq \
    &> logs/fastqc/$name.log
done

# 2. Bowtie2
mkdir -p results/bamfiles logs/bowtie2_index logs/bowtie2

bowtie2-build \
    $genome_fasta \
    bowtie2_index &> logs/bowtie2_index.log

cat $data | tail -n +2 | while read name group rep fastq; do
  (bowtie2 -p 4 \
      -x bowtie2_index \
      -U $fastq_dir/$fastq \
      -p $threads \
      | samtools view -bh - \
      | samtools sort -o results/bamfiles/$name.sorted.bam -) \
      2> logs/bowtie2/$name.log
done

# 3. Picard y samtools
mkdir -p logs/picard

cat $data | tail -n +2 | cut -f1 | while read name; do
  picard MarkDuplicates \
      I=results/bamfiles/$name.sorted.bam \
      O=results/bamfiles/$name.markdup.bam \
      M=results/bamfiles/$name.picard.txt \
      &> logs/picard/$name.log
done

mkdir -p logs/samtools_view

cat $data | tail -n +2 | cut -f1 | while read name; do
  samtools view -bh \
    -F 1024 -F 4 -q 5 \
    -@ $threads \
    results/bamfiles/$name.markdup.bam \
    > results/bamfiles/$name.clean.bam \
    2> logs/samtools_view/$name.log
done

cat $data | tail -n +2 | cut -f1 | while read name; do
  samtools index results/bamfiles/$name.clean.bam
done

# 4. bamCoverage y bigwigCompare
mkdir -p results/bigwigs logs/bamCoverage

cat $data | tail -n +2 | cut -f1 | while read name; do
  bamCoverage \
    -b results/bamfiles/$name.clean.bam \
    --normalizeUsing CPM \
    --extendReads 200 \
    -o results/bigwigs/$name.cpm.bw \
    -p $threads \
    &> logs/bamCoverage/$name.cpm.log
done

mkdir -p logs/bigwigCompare

cat $data | tail -n +2 | cut -f1 | grep -v $control | while read name; do
  bigwigCompare \
    -b1 results/bigwigs/$name.cpm.bw \
    -b2 results/bigwigs/$control.cpm.bw \
    -o results/bigwigs/${name}_over_${control}.log2ratio.bw \
    &> logs/bigwigCompare/${name}_over_${control}.log2ratio.log
done

# 5. MACS2
mkdir -p results/peak_calling logs/macs2

cat $data | tail -n +2 | cut -f1 | grep -v $control | while read name; do
  macs2 callpeak \
    -t results/bamfiles/$name.clean.bam \
    -c results/bamfiles/$control.clean.bam \
    -n $name \
    -g 119481543 \
    --keep-dup all \
    --outdir results/peak_calling \
    --nomodel --extsize 200 \
    &> logs/macs2/$name.log
done
```

Figura 5.5. Script empleado para el procesamiento de las lecturas del ChIP-seq desde el control de calidad hasta la llamada de picos.

```bash
#!/bin/bash

# Este script lleva a cabo la interseccion entre picos de varias muestras,
# su anotacion, filtrado y la busqueda de motivos enriquecidos

mkdir -p results/peak_analysis

# Interseccion entre picos
cat results/peak_calling/*_peaks.narrowPeak | grep -v pooled \
  | sortBed \
  | mergeBed -i - -c 4 -o collapse \
  | awk -v OFS="\t" '{ print $1, $2, $3, "rg_peak_"NR, $4 }' \
  > results/peak_analysis/rg_peaks.bed

# Summits de la interseccion entre picos
bedtools intersect \
  -a results/peak_analysis/rg_peaks.bed \
  -b results/peak_calling/rg_pooled_summits.bed \
  > results/peak_analysis/rg_summits.bed

# Filtrar los comunes a 2/3 replicas
python3 process_peaks.py
python3 filter_summits.py

# Anotar los picos con HOMER
mkdir -p logs/homer

annotatePeaks.pl rg_peaks_23.bed tair10 \
  > resuls/peak_analysis/rg_peaks_23.annotation.tsv \
  2> logs/homer/annotatePeaks.log

# Buscar motivos de ADN en los summits
findMotifsGenome.pl rg_summits_23.bed tair10 \
  results/peak_analysis/motifs \
  &> logs/homer/findMotifsGenome.log
```

Figura 5.6. Script empleado para el el análisis de los picos. Lleva a cabo la intersección entre picos de varias muestras, su anotación, filtrado y la búsqueda de motivos enriquecidos.

```python
#!/usr/bin/env python3

"""
Este script sirve para procesar la interseccion entre picos.
La salida de bedtools merge se transforma en un fichero TSV
con el numero de picos, muestras y una columna para cada muestra
indicando si contiene el pico (True) o no (False)
"""

def main():
    in_file = open("results/peak_analysis/rg_peaks.bed", "r")
    out_file = open("results/peak_analysis/rg_peaks.tsv", "w")
    sample_names = ["rg_rep1", "rg_rep2", "rg_rep3"]

    header = "\t".join(
        ["peak_id", "peaks", "num_peaks", "num_samples"] + sample_names
    )
    out_file.write(header + "\n")

    for line in in_file:
        fields = line.strip().split("\t")

        peaks = fields[4]
        samples = [sample in peaks for sample in sample_names]
        num_peaks = len(peaks.split(","))
        num_samples = sum(samples)

        new_line = "\t".join(
            fields[3:5]
            + [str(num_peaks), str(num_samples)]
            + [str(sample) for sample in samples]
        )
        out_file.write(new_line + "\n")

    in_file.close()
    out_file.close()


if __name__ == "__main__":
    main()

```

Figura 5.7. Script empleado para procesar la intersección entre picos. La salida de bedtools merge se transforma en un fichero TSV con el numero de picos, muestras y una columna para cada muestra indicando si contiene el pico (True) o no (False).

```python
#!/usr/bin/env python3

"""Este programa convierte ficheros GTF a BED con las coordenadas de cada gen"""

import sys

def main():
    if len(sys.argv) < 3:
        print("gtf2bed.py <gtf in> <bed out>")
        exit()

    input_gtf = sys.argv[1]
    output_bed = sys.argv[2]

    gtf = open(input_gtf, "r")
    bed = open(output_bed, "w")

    for line in gtf:
        fields = line.strip().split("\t")

        if fields[2] == "gene":
            info = fields[8].split(";")
            gene = info[1].strip().split(" ")[1].strip('"')

            line_bed = "\t".join(
                [fields[0], fields[3], fields[4], gene, fields[5], fields[6]]
            )
            bed.write(line_bed + "\n")

    gtf.close()
    bed.close()

if __name__ == "__main__":
    main()

```

Figura 5.8. Script empleado para convertir ficheros GFT en ficheros BED con las coordenadas de cada gen.

```python
#!/usr/bin/env python3

"""
Este script selecciona los picos presentes en al menos 2 replicas
y escribe nuevos ficheros con las regiones y los summits de estos
"""

def filter_and_write(input, output, peaks):

    input_file = open(input, "r")
    output_file = open(output, "w")

    for line in input_file:
        fields = line.strip().split("\t")

        if fields[3] in peaks:
            output_file.write(line)

    input_file.close()
    output_file.close()


def main():
    num_reps = 2

    selected_peaks = set()

    with open("results/peak_analysis/rg_peaks.tsv") as file:
        for line in file:
            fields = line.strip().split("\t")

            if fields[3] >= num_reps:
                selected_peaks.add(fields[0])


    filter_and_write(
        input="results/peak_analysis/rg_peaks.bed",
        output="results/peak_analysis/rg_peaks_23.bed",
        peaks=selected_peaks
    )

    filter_and_write(
        input="results/peak_analysis/rg_summits.bed",
        output="results/peak_analysis/rg_summits_23.bed",
        peaks=selected_peaks
    )


if __name__ == "__main__":
    main()

```

Figura 5.9. Script empleado para seleccionar los picos presentes en al menos 2 de las 3 réplicas del ChIP-seq y escribir nuevos ficheros con las regiones y los summits de estos.

```python
import sys    #rg_peaks.tsv

import matplotlib.pyplot as plt
from matplotlib_venn import venn3

input_file = sys.argv[1]

peaks_rep1 = set()
peaks_rep2 = set()
peaks_rep3 = set()

peaks_rep1_file = open("peaks_rep1.txt", "w")
peaks_rep2_file = open("peaks_rep2.txt", "w")
peaks_rep3_file = open("peaks_rep3.txt", "w")

with open(input_file) as f:
    f.readline()
    for line in f:

        fields = line.strip().split("\t")
        name = fields[0]
        rep1, rep2, rep3 = fields[2:5]


        if rep1 == "True":
            peaks_rep1.add(name)
            peaks_rep1_file.write(name + "\n")

        if rep2 == 'True':
            peaks_rep2.add(name)
            peaks_rep2_file.write(name + "\n")

        if rep3 == 'True':
            peaks_rep3.add(name)
            peaks_rep3_file.write(name + "\n")


venn3(
    [peaks_rep1, peaks_rep2, peaks_rep3],
    set_labels=["RGA rep1", "RGA rep2", "RGA rep3"],

plt.show()
```

Figura 5.10. Script empleado para la representación de un diagrama de Venn que muestra la intersección entre los picos de las diferentes réplicas del ChIP-seq.

```
tracks=$(ls *log2cpm.bw)


computeMatrix scale-regions \
  -R genes.bed \
  -S $tracks \
  -a 2000 -b 2000 -m 2000 \
  --missingDataAsZero --skipZeros \
  -o matrix_2000.gz \
  -p 3

```

Figura 5.11. Script empleado para la representación de un heatmap que distribución genómica de RGA en los genes diana.

```python
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""
#import numpy as np

import matplotlib.pyplot as plt

ann = {
    "promoter-TSS":0,
    "TTS":0,
    "Intergenic":0,
    "exon":0,
    "intron":0

}

with open('rg_peaks_23.annotation.tsv') as f:
    f.readline()
    for line in f:
        fields = line.strip().split('\t')
        info = fields[7].split(" ")[0]

        ann[info] += 1


sns.palplot (sns.color_palette ("hls", 9))

plt.pie(
        x=ann.values(),
        labels=ann.keys(),
        autopct="%.2f%%",


plt.show()

```

Figura 5.12. Script empleado para anotación de los picos en las diferentes categorías genómicas y su representación.