Bash Script

for SAM File Alignment Analysis
using an Assembly report

Muhammad Umer Hussain Kousar

TABLE OF CONTENTS

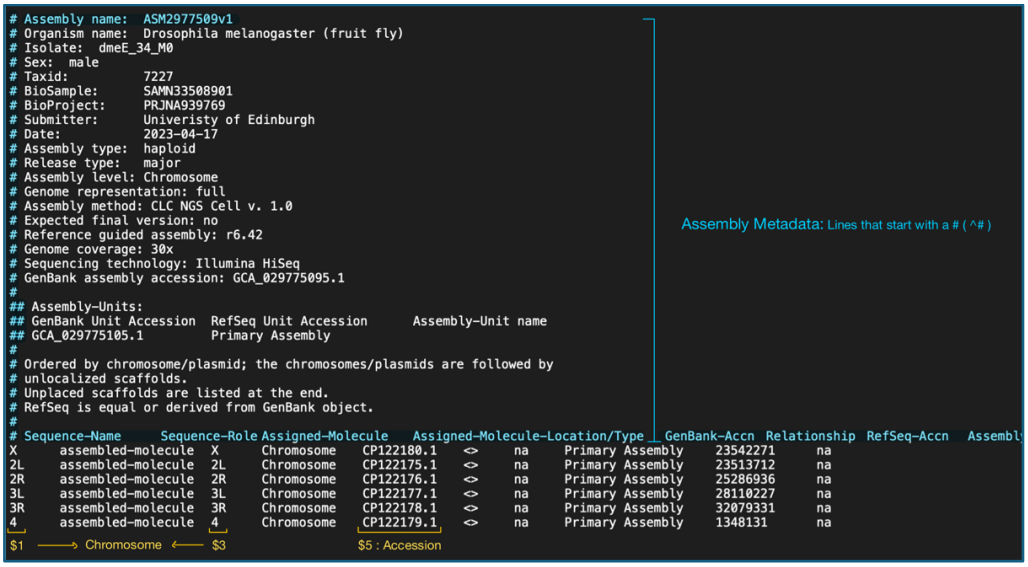
INPUT

This Bash script is designed to summarize sequence alignment results from one or more SAM (Sequence Alignment/Map) files and map those results to chromosomal locations using an Assembly report as a reference.

Files format

- Assembly report

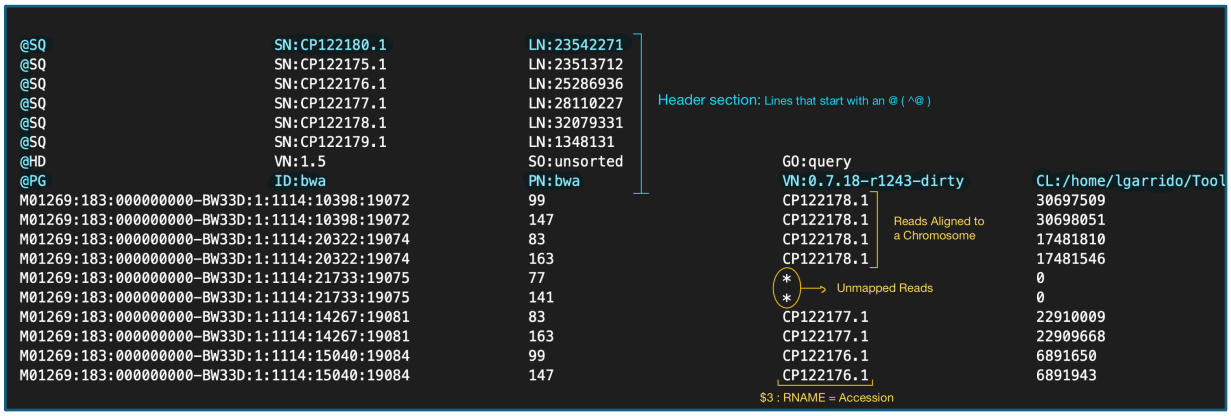
This is an NCBI-style genome assembly report that provides metadata for an organism's genomic DNA sequence — *in this case Drosophila melanogaster*. It includes mappings between chromosomes (e.g. 3R) and their corresponding GenBank accession numbers (e.g. CP122178.1). The key sections for this script are column 1 (chromosome) and column 5 (accessions).




```
# Assembly name:  ASM2977509v1
# Organism name:  Drosophila melanogaster (fruit fly)
# Isolate:  dmeE_34_M0
# Sex:  male
# Taxid:          7227
# BioSample:      SAMN33508901
# BioProject:     PRJNA939769
# Submitter:      Univeristy of Edinburgh
# Date:           2023-04-17
# Assembly type:  haploid
# Release type:   major
# Assembly level: Chromosome
# Genome representation: full
# Assembly method: CLC NGS Cell v. 1.0
# Expected final version: no
# Reference guided assembly: r6.42
# Genome coverage: 30x
# Sequencing technology: Illumina HiSeq
# GenBank assembly accession: GCA_029775095.1
#
## Assembly-Units:
## GenBank Unit Accession  RefSeq Unit Accession      Assembly-Unit name
## GCA_029775105.1         Primary Assembly
#
# Ordered by chromosome/plasmid; the chromosomes/plasmids are followed by
# unlocalized scaffolds.
# Unplaced scaffolds are listed at the end.
# RefSeq is equal or derived from GenBank object.
#
# Sequence-Name      Sequence-Role Assigned-Molecule   Assigned-Molecule-Location/Type  GenBank-Accn  Relationship  RefSeq-Accn   Assembly
X      assembled-molecule  X      Chromosome    CP122180.1    <>     na     Primary Assembly    23542271      na
2L     assembled-molecule  2L     Chromosome    CP122175.1    <>     na     Primary Assembly    23513712      na
2R     assembled-molecule  2R     Chromosome    CP122176.1    <>     na     Primary Assembly    25286936      na
3L     assembled-molecule  3L     Chromosome    CP122177.1    <>     na     Primary Assembly    28110227      na
3R     assembled-molecule  3R     Chromosome    CP122178.1    <>     na     Primary Assembly    32079331      na
4      assembled-molecule  4      Chromosome    CP122179.1    <>     na     Primary Assembly    1348131       na
```

- SAM file

This is a tab-delimited file that stores records of DNA sequencing reads mapped against a reference genome. The Header section provides metadata about the alignment process, while each non-header line represents an individual read, either aligned to a reference sequence (accession / chromosome) or not (*). This information is indicated on column 3.




```
@SQ                     SN:CP122180.1           LN:23542271
@SQ                     SN:CP122175.1           LN:23513712
@SQ                     SN:CP122176.1           LN:25286936
@SQ                     SN:CP122177.1           LN:28110227
@SQ                     SN:CP122178.1           LN:32079331
@SQ                     SN:CP122179.1           LN:1348131
@HD                     VN:1.5                  SO:unsorted
@PG                     ID:bwa                  PN:bwa          GO:query
M01269:183:000000000-BW33D:1:1114:10398:19072   99      CP122178.1      30697509
M01269:183:000000000-BW33D:1:1114:10398:19072   147     CP122178.1      30698051
M01269:183:000000000-BW33D:1:1114:20322:19074   83      CP122178.1      17481810
M01269:183:000000000-BW33D:1:1114:20322:19074   163     CP122178.1      17481546
M01269:183:000000000-BW33D:1:1114:21733:19075   77      *               0
M01269:183:000000000-BW33D:1:1114:21733:19075   141     *               0
M01269:183:000000000-BW33D:1:1114:14267:19081   83      CP122177.1      22910009
M01269:183:000000000-BW33D:1:1114:14267:19081   163     CP122177.1      22909668
M01269:183:000000000-BW33D:1:1114:15040:19084   99      CP122176.1      6891650
M01269:183:000000000-BW33D:1:1114:15040:19084   147     CP122176.1      6891943
```

Terminal input

The script on the Terminal should be executed as follows:

```
./RUScript.sh <file1.sam> … <fileN.sam> <assembly_report>
```

The input provided is interpreted as follows:

- All arguments except the last are treated as a SAM files.
- The last argument provided is treated as the Assembly report.

```
# Input files for the script

SAM_FILES=("${@:1:$#-1}") # all parameters before the last one are treated as SAM files
ASSEMBLY_REPORT=("${@: -1}") # the final parameter must be the assembly report.
```

If the input is incorrect or incomplete (e.g. missing files, no assembly report as the last argument, or invalid SAM files), the script will display an error message on the Terminal and quit the process with `exit 1`.

```
# Usage Instructions and Exit script if given wrong input

check_input() {

  # check number of parameters

  if [ "$#" -lt 2 ]; then
    echo
    echo " - Hey, I need at least One Sam file and Exactly One Assembly report as the LAST parameter!"
    echo " - Run it like this: $0 <file1.sam> <file2.sam> ... <assembly_report>"
    echo
    exit 1
  fi

  # check last parameter

  if [[ "$(echo "$ASSEMBLY_REPORT" | tr '[:upper:]' '[:lower:]')" == *.sam ]]; then
    echo
    echo " - The last parameter MUST be an Assembly report"
    echo " - Run it like this: $0 <file1.sam> <file2.sam> ... <assembly_report>"
    echo
    exit 1
  fi

  # check SAM files

  for f in "${SAM_FILES[@]}"; do
    file_name=$(echo "$f" | tr '[:upper:]' '[:lower:]')

    if [[ "$file_name" != *.sam ]]; then
      echo
      echo " - All parameters except the last MUST be SAM files!"
      echo
      exit 1
    fi
  done

}

check_input "$@"
```

e.g. Terminal output when the number of arguments is insufficient

```
bash-3.2$ ./RUScript.sh

 - Hey, I need at least One Sam file and Exactly One Assembly report as the LAST parameter!
 - Run it like this: ./RUScript.sh <file1.sam> <file2.sam> ... <assembly_report>

bash-3.2$ ./RUScript.sh Sam_file.sam

 - Hey, I need at least One Sam file and Exactly One Assembly report as the LAST parameter!
 - Run it like this: ./RUScript.sh <file1.sam> <file2.sam> ... <assembly_report>

bash-3.2$ ./RUScript.sh assembly.txt

 - Hey, I need at least One Sam file and Exactly One Assembly report as the LAST parameter!
 - Run it like this: ./RUScript.sh <file1.sam> <file2.sam> ... <assembly_report>
```

e.g. Terminal output when the last argument is not an assembly report

```
bash-3.2$ ./RUScript.sh Sam_file.sam Sam_file_test.sam

  - The last parameter MUST be an Assembly report
  - Run it like this: ./RUScript.sh <file1.sam> <file2.sam> ... <assembly_report>

 bash-3.2$ ./RUScript.sh Sam_file.sam assembly.txt Sam_file_test.sam

  - The last parameter MUST be an Assembly report
  - Run it like this: ./RUScript.sh <file1.sam> <file2.sam> ... <assembly_report>
```

e.g. Terminal output when one or more SAM files have an invalid format

```
bash-3.2$ ./RUScript.sh Sam_file.sam RUScript.sh assembly.txt

 - All parameters except the last MUST be SAM files!

bash-3.2$ ./RUScript.sh Sam_file.sam assembly.txt RUScript.sh assembly.txt

 - All parameters except the last MUST be SAM files!
```

On the other hand, if the input is valid and all files are processed successfully, the following message will be displayed on the Terminal:

```
# Indicate succesful execution of the script on the Terminal
echo
echo " - Script executed successfully ✅"
echo " - To check the output, run: cat output.txt"
echo
```

e.g. Terminal output when given correct input

```
bash-3.2$ ./RUScript.sh Sam_file.sam assembly.txt

 - Script executed successfully ✅
 - To check the output, run: cat output.txt

bash-3.2$ ./RUScript.sh Sam_file.sam Sam_file_test.sam assembly.txt

 - Script executed successfully ✅
 - To check the output, run: cat output.txt
```

OUTPUT

Before starting the analysis, any output files used for appending information throughout the script are cleared to prevent incorrect results.

```
# ------- OUTPUT -------

OUTPUT_FILE="output.txt" # output file for the script

  # clear files at the start

  > "$OUTPUT_FILE"
  > acc_tf # temporary file for accessions from each SAM file
```

Accession to Chromosome map

The script processes the assembly report by extracting lines that do not start with `#`, retrieving the accession number (column 5) and its corresponding chromosome (column 1). These results are then sorted in ascending order by accession number and saved in the file `< acc_to_chr_tf >`.

```
# Accession - Chromosome map -> temporary file

awk '!/^#/ {print $5 "\t" $1}' "$ASSEMBLY_REPORT" | sort > acc_to_chr_tf
```

Content of < acc_to_chr_tf >

```
bash-3.2$ cat acc_to_chr_tf
CP122175.1      2L
CP122176.1      2R
CP122177.1      3L
CP122178.1      3R
CP122179.1      4
CP122180.1      X
```

SAM file Analysis

```
  # SAM file analysis

for sam_file in "${SAM_FILES[@]}"; do

    # Count number of reads processed per SAM file (non-header lines)
    reads=$(grep -vc "^@" "$sam_file")

    # Count number of aligned reads per SAM file (non-header lines with RNAME ≠ "*")
    aligned_reads=$(awk '!/^@/ && $3 != "*"' "$sam_file" | wc -l)

    # Totals across all SAM files
    (( total_reads += reads ))
    (( total_aligned_reads += aligned_reads ))

    # Append aligned reads accessions from each SAM file -> temporary file
    awk '!/^@/ && $3 != "*" {print $3}' "$sam_file" >> acc_tf

done
```

- Number of Reads Processed

To determine the total number of reads, we need to find `grep` and count `-c` the number of non-header lines — i.e. lines that do not `-v` start with `@` — from each SAM file:

e.g. Count the number of reads in < Sam_file.sam >

```
bash-3.2$ grep -vc "^@" Sam_file.sam
368465
```

- Number of Aligned Reads

To count aligned reads, we need to count only non-header lines that are mapped to a reference — i.e. lines that do not contain `*` in column 3 (RNAME) — from each SAM file:

e.g. Count the number of aligned reads in < Sam_file.sam >

```
bash-3.2$ awk '!/^@/ && $3 != "*"' Sam_file.sam | wc -l
  345231
```

After that, we sum the number of total and aligned reads from all SAM files to obtain a global count.

- Number of Aligned Reads per Accession / Chromosome

To count the number of aligned reads per accession / chromosome, we first need to determine how many times each accession appears in each SAM file. For this, we first generate a temporary file `< acc_tf >` containing all accessions in the order they appear within the SAM file. Note that accessions extracted from each SAM file are appended `>>` to those from the previously processed files.

Content of < acc_tf >

```
bash-3.2$ less acc_tf
CP122180.1
CP122178.1
CP122178.1
CP122180.1
CP122180.1
CP122177.1
CP122177.1
```

Next, we `sort` the `< acc_tf >` file and count the occurrences of each accession `uniq -c`. Then, we swap the columns so that the accession becomes the first column (*$2 → $1*) and the count becomes the second one (*$1 → $2*), saving the results in a new temporary file called `< total_acc_count_tf >`. This will be essential for generating the final output table associating each accession with its corresponding chromosome and read count, using the `join` command.

```
# Count number of aligned reads for each accession across all SAM files -> temporary file
sort acc_tf | uniq -c | awk '{print $2 "\t" $1}' > total_acc_count_tf
```

Sorted Count of each accession in < acc_tf >

```
bash-3.2$ sort acc_tf | uniq -c
58436 CP122175.1
63826 CP122176.1
73734 CP122177.1
77924 CP122178.1
3720 CP122179.1
67591 CP122180.1
```

Formatted Acc-Count table in < total_acc_count_tf >

```
bash-3.2$ cat total_acc_count_tf
CP122175.1      58436
CP122176.1      63826
CP122177.1      73734
CP122178.1      77924
CP122179.1      3720
CP122180.1      67591
```

Format of Acc-Chr table in < acc_to_chr_tf >

```
bash-3.2$ cat acc_to_chr_tf
CP122175.1      2L
CP122176.1      2R
CP122177.1      3L
CP122178.1      3R
CP122179.1      4
CP122180.1      X
```

This latter file `< acc_to_chr_tf >` we generated earlier, is already sorted by accession to facilitate joining with read count data in `< total_acc_count_tf >`.

Output file

After completing the analysis, the final step is to print the results in the previously set output file `< output.txt >`.

```
# Final Output

{

  echo
  printf "%-3s %-3s %s\n" "===" "SAM FILES ALIGNMENT ANALYSIS" "==="
  echo
  echo
  printf "%-20s %s\n" "Total reads processed:" "$total_reads"
  printf "%-21s %s\n" "Aligned reads:" " $total_aligned_reads"
  echo
  echo
  printf "%-21s %-15s %s\n" "Accession" "Chromosome" "Aligned Reads"
  printf "%-21s %-15s %s\n" "---------------------" "---------------" "---------------"
  join -t $'\t' acc_to_chr_tf total_acc_count_tf | awk '{printf "%-21s %-15s %s\n", $1, $2, $3}'
  echo
  echo

} >> "$OUTPUT_FILE" 2>&1
```

To measure the execution time, we use a Bash built-in variable called `SECONDS`. It is set to `0` at the start of the script `SECONDS=0` and its value `$SECONDS` is called at the end of the script to determine the total runtime in seconds.

```
#!/bin/bash

SECONDS=0 # Set variable for timing
```

```
# Execution time of the script (in seconds)
printf "%-20s %s\n" "Total execution time:" "$SECONDS s" >> "$OUTPUT_FILE" 2>&1
```

Content of < output.txt >

```
bash-3.2$ cat output.txt

=== SAM FILES ALIGNMENT ANALYSIS ===


Total reads processed: 368465
Aligned reads:         345231


Accession             Chromosome      Aligned Reads
---------------------------------------------------
CP122175.1            2L              58436
CP122176.1            2R              63826
CP122177.1            3L              73734
CP122178.1            3R              77924
CP122179.1            4               3720
CP122180.1            X               67591


Total execution time: 6 s
```

Delete junk

Finally, all temporary files generated during the script are removed, as they are no longer needed.

```
rm -f acc_to_chr_tf acc_tf total_acc_count_tf # remove all temporary files
```