Below is a detailed README that explains how to use and chain together the three scripts:

1. coverage.py
2. log.py
3. clustering.py (the final script)

These scripts together perform a pipeline that:

1. Calculates coverage-based median windows around each nucleotide in a genome (looking for potential terminators).
2. Computes log2 fold changes from pairs of coverage files.
3. Clusters significant regions and annotates them with gene and sequence features.

**Pipeline Overview**

1. coverage.py
   * **Goal:** Load coverage files (forward/reverse) for a sample, normalize them to TPM, calculate 100-nt upstream and downstream median coverage for each nucleotide, then compute a ratio.
   * **Output**: Two .txt (TSV) files—one for forward coverage and one for reverse coverage. Each file has:
     + Coord
     + Forward\_Upstream\_Median, Forward\_Downstream\_Median, Forward\_Ratio *(if forward strand)*
     + OR Reverse\_Upstream\_Median, Reverse\_Downstream\_Median, Reverse\_Ratio *(if reverse strand)*
2. log.py
   * **Goal:** Take the coverage ratio files from coverage.py for two different conditions (e.g. wild-type vs. dRho mutant) and compute a log2 fold change for each position.
   * **Output:** Two .txt (TSV) files—one for forward strand differentials, one for reverse strand differentials. Each file has columns:
     + Coord
     + WT\_Ratio, dRho\_Ratio
     + Log2\_FC (log2 fold change)
     + Additional columns capturing upstream/downstream medians from both conditions.
3. clustering.py
   * **Goal:** Take the log2 fold change tables from log.py, filter by thresholds, cluster overlapping positions, merge close clusters into “pseudo-clusters,” and annotate them with gene/sequence info from a GFF.
   * **Output:** A single .csv file containing cluster-level annotations, including:
     + Start, End, Center, ratio/log2-FC stats
     + Gene/TSS/ATG annotation from the GFF
     + Upstream/Downstream sequences
     + Additional sequence-content features (like CT or G content in sliding windows).

**1. coverage.py**

**Description**

This script reads a forward and a reverse coverage .cov file (tab-delimited) that were generated externally (e.g. from DNAnexus). For each position in the genome:

* It takes 100 nucleotides upstream and 100 nucleotides downstream, computes the median coverage in each window, and stores those medians.
* Then computes a ratio:
  + For the forward strand: Upstream\_Median / Downstream\_Median
  + For the reverse strand: Downstream\_Median / Upstream\_Median
* Optionally normalizes the coverage to TPM (Transcripts Per Million) so that different files can be compared on the same scale.

**Inputs**

1. **Forward Coverage File** (.cov)  
   A tab-delimited file with columns like [chromosome, position, coverage\_value].
2. **Reverse Coverage File** (.cov)  
   Same format as the forward coverage file.
3. **Chromosome Length**  
   An integer indicating how many nucleotides make up the chromosome. Default in the script is 4639675 for *E. coli*K-12.
4. **Window Size**  
   Currently set to 100. You can adjust this if needed in the script.
5. **Hard-coded file paths**  
   In the script, the forward coverage file (forward\_cov\_path) and reverse coverage file (reverse\_cov\_path) are specified. You may need to **edit** these paths to point to your actual .cov files.

**Outputs**

* Two .txt files (tab-separated), by default named:
  + dRho\_1261\_Ara\_92\_101\_102\_m14\_M30\_fwd\_coverage.txt
  + dRho\_1261\_Ara\_92\_101\_102\_m14\_M30\_rev\_coverage.txt
* Each file contains:
  + Coord: The 1-based genomic coordinate.
  + Forward\_Upstream\_Median / Forward\_Downstream\_Median / Forward\_Ratio (if forward).
  + OR Reverse\_Upstream\_Median / Reverse\_Downstream\_Median / Reverse\_Ratio (if reverse).

**How to Run**

bash

CopyEdit

python coverage.py

* Make sure you edit coverage.py so that:
  + forward\_cov\_path and reverse\_cov\_path point to the correct coverage files.
  + chromosome\_length matches your genome.
  + window\_size matches your desired upstream/downstream distance.

**2. log.py**

**Description**

This script is designed to run after coverage.py. It takes two sets of coverage-ratio files (for forward and reverse strands), one for wild-type (WT) and one for dRho (or any condition you’re comparing to WT). Then it:

1. Loads those files.
2. Merges them by coordinate.
3. Calculates the log2 fold change (-np.log2(dRho\_Ratio / WT\_Ratio)).
4. Adds some additional columns about upstream/downstream medians from both files.

**Inputs**

1. **WT Forward Coverage** (TSV)  
   Generated by coverage.py, containing columns Coord, Forward\_Upstream\_Median, etc.
2. **WT Reverse Coverage** (TSV)  
   Same as above, for the reverse strand.
3. **dRho (or experimental) Forward Coverage** (TSV)  
   Generated by coverage.py for your dRho or mutant condition.
4. **dRho (or experimental) Reverse Coverage** (TSV)

The script has hard-coded file paths:

python

CopyEdit

wt\_forward\_df = pd.read\_csv('1325\_WT\_199\_200\_201\_fwd\_coverage.txt', sep='\t')

dRho\_forward\_df = pd.read\_csv('dRho\_1261\_Ara\_92\_101\_102\_m14\_M30\_fwd\_coverage.txt', sep='\t')

...

Edit these to point to your actual coverage output files from coverage.py.

**Outputs**

* Two TSV files containing log2 fold change across the genome:
  1. 1325\_WT\_199\_200\_201\_vs\_dRho\_1261\_Ara\_92\_101\_102\_m14\_M30\_forward\_strand\_differential\_table.txt
  2. 1325\_WT\_199\_200\_201\_vs\_dRho\_1261\_Ara\_92\_101\_102\_m14\_M30\_reverse\_strand\_differential\_table.txt
* Each file has columns:
  1. WT\_Ratio
  2. dRho\_Ratio
  3. Log2\_FC
  4. WT\_Forward\_Upstream\_Median, WT\_Forward\_Downstream\_Median, etc.  
     (The exact columns differ slightly for forward vs. reverse.)

**How to Run**

bash

CopyEdit

python log.py

* Ensure the file names in wt\_forward\_df = pd.read\_csv(...) etc. match what was generated by coverage.py.

**3. clustering.py**

**Description**

This final script performs **filtering, clustering, merging, and annotation** steps:

1. **Filter** the log2 fold change data for each strand based on thresholds (e.g. Log2\_FC >= 1.8, WT\_Ratio >= 4, etc.).
2. **Identify Overlaps** and group close positions into “pseudo-clusters.”
3. **Merge** clusters that are within 60 nt of each other.
4. **Calculate cluster statistics** (start, end, max ratio, max log2 FC, etc.).
5. **Annotate each cluster** with:
   * **Gene** information (using a local GFF and gffutils),
   * **Upstream/Downstream sequences** extracted from the reference genome (via Biopython),
   * **Closest TSS**/**ATG** (start codon) information using a TSS CSV,
   * Additional sequence composition features (C/T content, G content, etc.).

**Inputs**

1. **Forward Log2-FC File**  
   Produced by log.py, e.g. 1325\_WT\_199\_200\_201\_vs\_dRho\_1261\_Ara\_92\_101\_102\_m14\_M30\_forward\_strand\_differential\_table.txt
2. **Reverse Log2-FC File**  
   Produced by log.py.
3. **Reference Genome** in FASTA format, e.g. NC\_000913.2.fasta
4. **GFF File** for annotation, e.g. NC\_000913.2.gff3
5. **GFF Database** (.db file for gffutils), e.g. NC\_000913.2.db
   * If it doesn’t exist, the script will create it.
6. **TSS.csv** (Start sites)  
   Contains columns like TSS\_site and TSS\_direction to help find the closest TSS.

**Outputs**

* A single CSV file (e.g. combined\_strands\_1325\_WT\_199\_200\_201\_vs\_dRho\_1261\_Ara\_92\_101\_102\_m14\_M30\_clusters\_identified\_100.csv) with the final cluster annotations.
* Columns include cluster coordinates, ratio/log2FC stats, gene annotation, TSS/ATG data, and various sequence features.

**How to Run**

bash

CopyEdit

python clustering.py

* Again, check the top of the script for **hard-coded paths** to the forward\_file, reverse\_file, gff\_file, db\_file, tss\_file, etc. You will need to edit these to match your local filenames.

**Dependencies**

All scripts require the following Python packages:

* Python 3.7+
* NumPy (e.g. pip install numpy)
* pandas (e.g. pip install pandas)
* BioPython (e.g. pip install biopython)
* gffutils (e.g. pip install gffutils)
* Optionally, BCBio.GFF is used in some contexts to parse GFF, but the main functionality is via gffutils.

**Typical Workflow Example**

1. **Generate .cov files** (forward & reverse) for your sample (WT or mutant) from DNAnexus or another coverage tool.
2. **Run coverage.py** for each sample (WT, dRho, etc.):

bash

CopyEdit

python coverage.py

This produces, for example:

* + wt\_fwd\_coverage.txt / wt\_rev\_coverage.txt
  + dRho\_fwd\_coverage.txt / dRho\_rev\_coverage.txt

1. **Run log.py** using the outputs from step 2 to get log2 FC for forward & reverse:

bash

CopyEdit

python log.py

This produces:

* + wt\_vs\_dRho\_forward\_strand\_differential\_table.txt
  + wt\_vs\_dRho\_reverse\_strand\_differential\_table.txt

1. **Run clustering.py** with the log2 FC files from step 3, a reference genome FASTA, a GFF, and a TSS CSV:

bash

CopyEdit

python clustering.py

This produces a final .csv with cluster annotations.

**Troubleshooting & Tips**

1. Check Hard-Coded Paths
   * In each script, you’ll see variables like forward\_cov\_path, reverse\_cov\_path, or direct pd.read\_csv(...)calls. Ensure these match your file names.
2. Adjust Window Sizes
   * window\_size = 100 in coverage.py and likewise in clustering.py for sequence extraction. You can change it to 50, 200, etc. as needed.
3. Edit Filtering Criteria (in clustering.py)
   * Currently: Log2\_FC >= 1.8, WT\_Ratio >= 4, etc. Modify to your threshold preferences.
4. Missing NC\_000913.2.db
   * If it doesn’t exist, gffutils will create it. Check for memory/time constraints if your GFF is large.
5. Edge Cases
   * If your coverage is too low, you might get NaN or Inf in ratios. The scripts attempt to handle that by setting them to 0 or ignoring them in clustering.