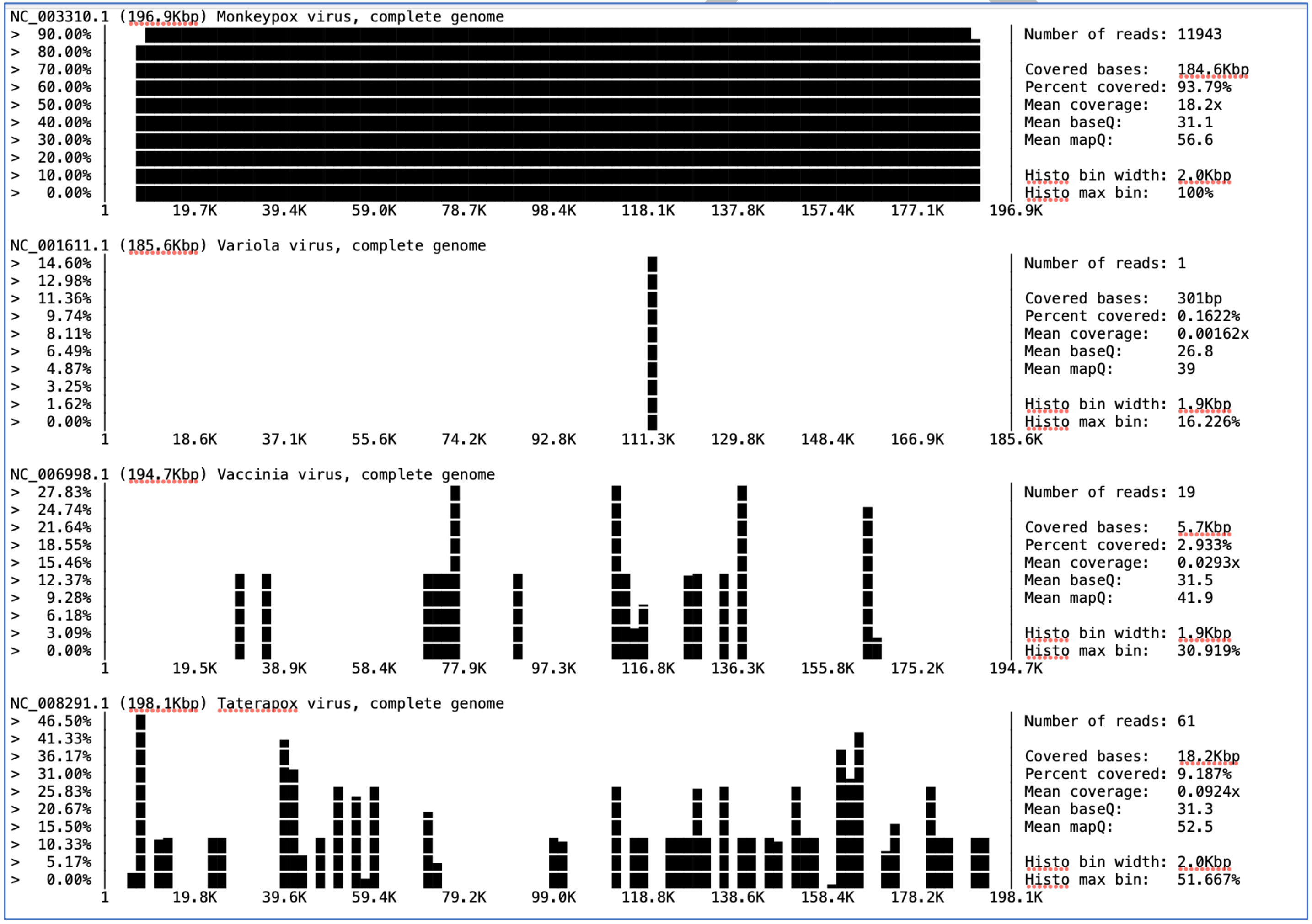
Threat Agnostic Sentinel Surveillance: Informatics Development

Confidence Scoring for Metagenomic Pathogen Detection

Establishing Confidence Scoring Methodologies:

TaxTriage has included an experimental confidence score for each organism identified within a sample represented as of TaxTriage v2.0.8 [documentation here]. This version of the score includes five different variables to assess likelihood of an organism identified using a metagenomic classifier (e.g. Kraken2, DIAMOND) being well represented by the underlying data after direct genome alignment (e.g. Bowtie2, minimap2). The final score is a weighted sum of several factors:

In order to agnostically identify groups of references that hold similar or multi-mapped read assignments, we must identify patterns and clustered traits attribute to potential off-target or false positive alignments. For example, in many Orthopox virus tests, a multitude of genera are found that are not truly found in, say, a Monkeypox sample. That is, for a given in-silico-generated dataset, we would see histograms like:



```
NC_003310.1 (196.9Kbp) Monkeypox virus, complete genome
>  90.00% |                                                       | Number of reads: 11943
>  80.00% |                                                       |
>  70.00% |                                                       | Covered bases:   184.6Kbp
>  60.00% |                                                       | Percent covered: 93.79%
>  50.00% |                                                       | Mean coverage:   18.2x
>  40.00% |                                                       | Mean baseQ:      31.1
>  30.00% |                                                       | Mean mapQ:       56.6
>  20.00% |                                                       |
>  10.00% |                                                       | Histo bin width: 2.0Kbp
>   0.00% |                                                       | Histo max bin:   100%
          1      19.7K    39.4K    59.0K    78.7K    98.4K    118.1K   137.8K   157.4K   177.1K   196.9K

NC_001611.1 (185.6Kbp) Variola virus, complete genome
>  14.60% |                                                       | Number of reads: 1
>  12.98% |                                                       |
>  11.36% |                                                       | Covered bases:   301bp
>   9.74% |                                                       | Percent covered: 0.1622%
>   8.11% |                                                       | Mean coverage:   0.00162x
>   6.49% |                                                       | Mean baseQ:      26.8
>   4.87% |                                                       | Mean mapQ:       39
>   3.25% |                                                       |
>   1.62% |                                                       | Histo bin width: 1.9Kbp
>   0.00% |                                                       | Histo max bin:   16.226%
          1      18.6K    37.1K    55.6K    74.2K    92.8K    111.3K   129.8K   148.4K   166.9K   185.6K

NC_006998.1 (194.7Kbp) Vaccinia virus, complete genome
>  27.83% |                                                       | Number of reads: 19
>  24.74% |                                                       |
>  21.64% |                                                       | Covered bases:   5.7Kbp
>  18.55% |                                                       | Percent covered: 2.933%
>  15.46% |                                                       | Mean coverage:   0.0293x
>  12.37% |                                                       | Mean baseQ:      31.5
>   9.28% |                                                       | Mean mapQ:       41.9
>   6.18% |                                                       |
>   3.09% |                                                       | Histo bin width: 1.9Kbp
>   0.00% |                                                       | Histo max bin:   30.919%
          1      19.5K    38.9K    58.4K    77.9K    97.3K    116.8K   136.3K   155.8K   175.2K   194.7K

NC_008291.1 (198.1Kbp) Taterapox virus, complete genome
>  46.50% |                                                       | Number of reads: 61
>  41.33% |                                                       |
>  36.17% |                                                       | Covered bases:   18.2Kbp
>  31.00% |                                                       | Percent covered: 9.187%
>  25.83% |                                                       | Mean coverage:   0.0924x
>  20.67% |                                                       | Mean baseQ:      31.3
>  15.50% |                                                       | Mean mapQ:       52.5
>  10.33% |                                                       |
>   5.17% |                                                       | Histo bin width: 2.0Kbp
>   0.00% |                                                       | Histo max bin:   51.667%
          1      19.8K    39.6K    59.4K    79.2K    99.0K    118.8K   138.6K   158.4K   178.2K   198.1K
```

In this case, Monkeypox is a true-positive spike-in generated from insilicoseq using the miseq model. Following alignment with minimap2 against a list of references that include ALL orthopox representative species from Refseq, we see some potential off-target sites that are shared or conserved amongst the genera. In terms of reporting, this proves problematic as they are certainly considered pathogens but must be accounted for and removed to not give an improper reflex response. Using a variety of methods, agnostic of genera or taxonomy, we can derive a rough analytical technique for "binning" these alignment patterns into groups based on multi-reference mapped reads as well as trends in read depth across each genome belonging to said group.

Confidence Scoring Methodology v2.0:

In order to refine our confidence metric analysis and improve on removal of false positive hits, we utilize a multi-step process using alignment output from popular aligners such as minimap2. Other utilities, such as samtools and clustering techniques are then used to identify regional trends in alignments and bin or group organism alignments based on a variety of factors, regardless or agnostic of their taxonomies.

The confidence metric analysis has been merged into the primary releases under tag: 2.0.0

Reads are first mapped with minimap2 (paired or single-end) with a minimum MAPQ of 10. From there, the alignments are then passed through bedtools genomecov (using the BAM file) to identify the range of depths across all chromosomes/contigs for all references aligned from the original top hits and pathogens data sheet.

A. Gini Coefficient (G) and AFDS (Average Fair Distribution Score)

The Gini coefficient measures the inequality in the distribution of sequencing depths across the genome positions:

- $\boldsymbol{G = 0}$ Perfect equality (uniform coverage across all positions).
- $\boldsymbol{G = 1}$ Maximum inequality (all coverage concentrated at a single position).

However, for simplicity, we inverse the typical scores to indicate that 1 is equal to perfect equality across the positions and 0 as inequal. The description of how alignments to references are calculated is described below.

Summary

1. Compute the histogram (H_{tr}) which is the distribution of depths and coverages for each reference
2. Calculate the G (Gini coefficient) from the $\boldsymbol{H_{tr}}$
3. Determine a scaling factor based on the genome length: $L, L_{base}, L_{max}, r_f$
4. Compute dispersion factor: $1 + \beta D$
5. Multiply these together to get the AFDS (Average Fair Distribution Score)

Transforming the depth and coverage profiles into a histogram (H_{tr})

$$\boldsymbol{H_{tr}}(\boldsymbol{x}) = \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \mathbf{1}\{\boldsymbol{T}(\boldsymbol{d_i}) = \boldsymbol{x}\}(\boldsymbol{e_i} - \boldsymbol{s_i}) + \delta_{x,T(0)} \left(\boldsymbol{L} - \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} (\boldsymbol{e_i} - \boldsymbol{s_i})\right)$$

Where $regions = \{(s_i, e_i, d_i) \mid i = 1, \dots, n\}$

The goal here is to take into account all of the areas where an alignment occurs.

- $\delta_{x\,T(0)}$ is the is equal to 1 for any depth == 0 and 0 for empty positions (no coverage)
- $\mathbf{1}\{\boldsymbol{T}(\boldsymbol{d_i}) = \boldsymbol{x}\}$ is equal to 1 if d the depth of the current position is the same as the previous

Calculating the full fair distribution score

$$Score_{TASS} = \min\left\{1, \left[\alpha\sqrt{1 - G}\right] \cdot \left[1 + r_f \cdot \log10\left(max\left\{\frac{min\{L, L_{max}\}}{L_{base}}, 1\right\}\right)\right] x\,[1 + \beta D]\right\}$$

Broken down, we have

We can express the overall calculation as a single equation. Let

- G be the raw Gini coefficient computed from the transformed coverage histogram (H_{tr})
- L be the genome length
- L_{max} be the maximum allowed length (max length)
- L_{base} be the baseline value
- r_f be the reward factor
- α be the transformation parameter
- β be the weight for the dispersion factor
- D be the positional dispersion factor

First, we take into consideration the histogram information from $\boldsymbol{H_{tr}}$ to calculate the gini coefficient based on the depth profiles

$$G = \frac{1}{2N^2\mu_T} \sum_{d} \sum_{d_i} H_{tr}(d) H_{tr}(d') T(d) - T(d')$$

Where:

- $N = \sum_d H_{tr}(d)$ is the sum of all bases
- $\mu = \frac{\sum_d H_{tr}(d)}{N}$ is the mean of the coverage depth profiles

We then calculate the Log of the Gini Coefficient (G) (inverse)

$$g_{log} = \alpha\sqrt{1 - G}$$

Next, we cap the genome length at: $L' = \min\{L, L_{max}\}$

And then calculate the ratio of lengths as: $r = \max\left\{\frac{L'}{L_{max}}, 1\right\}$

$$\text{Scaling Factor} = 1 + r_f \; x \; log_{10}(r)$$

Which is used to penalize a poor g_{log} for small genomes and make a greater reward for less inequal depths and coverages for longer genomes.

Next, we use the Dispersion factor: $1 + \beta D$, which is designed to dampen penalties for more spread-out regions statically (regardless of the reference size)

B. MinHash Similarity (S_{Comp}) Using Sourmash

In order to determine the minimizers i.e. the sets of unique k-mers hashed using the methods defined in sourmash, we need to determine how “similar” each of the reads are to all other minimizers present in a dataset. For example, in our orthopox example, can we define how many hashes are “matched” in higher similarity to the ground truth monkeypox versus the false positives?

However, we must first identify what regions are to be compared in the dataset. The comparisons required for all reads would be subject to **O(N^2)** which is not feasible for millions of reads as this could take upwards of 12 hours total. As a result, we must condense the information into a regional merge using bedtools information. By passing the information from the *bedtools genomcov* subcommand we can identify what the reference NT’s that are shared with all alignments. This is instead **O(M * N)** where M is the number of regions (those that all contain the same depth value). An example bedgraph can be seen here for some Orthopox viruses:

Reference	Start	End	Depth
NC_003310.1	190758	190761	6
NC_003310.1	190761	190762	5
NC_003310.1	190762	190769	4
NC_003310.1	190769	190771	2
NC_003310.1	190771	190773	1
NC_006998.1	68492	68793	1
NC_006998.1	81246	81547	1
NC_008291.1	19267	19288	1
NC_055230.1	94740	95041	1

While this process dramatically reduces the time complexity, the depth profiles can vary widely between each positions. In most bacteria alignments, we can see upwards of 10k unique regions per chromosome, which causes compute time to increase substantially. As a result, we employ a statistical method using either the variance of depth-skips, the gini coefficient (similar to the AFDS methodology), or by setting a threshold (static) value before a new region is defined.

During the following calculations, the sorted regions per reference are calculated using a buffer-style methodology. That is, as the first region is analyzed (first seen start-end and depth value), a buffer is initialized. Each of the statistical methods are then compare to the previously derived value until a set threshold is reached (more on each threshold in each function). If the threshold is reached, the buffer is saved as a merged region and a new one is created. The complexity of the buffer is highly reliant on sorting the regions as well, and will typically be only **O(n)** where n is the number of regions. A detailed look at how each of these are calculated are as follows:

Population variance of region and depth:

The slowest method but most precise:

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

Where $\bar{\mathbf{x}}$ is the mean of the depth profiles across the reference bedgraphs. This method also produces the most comparisons needed downstream i.e. less regional merges. The default threshold is 0.8 for regional buffer merges.

Gini Coefficient (default):

A slightly faster method, this is calculated as:

$$G = \frac{2\sum_{i=1}^{n} i\ x_{(i)}}{n\sum_{i=1}^{n} x_i} - \frac{n+1}{n}$$

This method takes into consideration all of the depths and calculates the disparity value. The default threshold is 0.8 (inequality score for the Gini coefficient).

Jump Method:

This method is the most simplistic, and derives the median "jump (c)" between regional depths as the threshold. This threshold is then compared like so:

$$\mathrm{c} = |x_{\text{current}} - x_{\text{last}}|$$

In short, the next region's depth is compared to the last region. If the jump is too large, then a buffer is saved/flushed and a new one is initialized. Ultimately, this is the quickest method and shows comparable results to the Gini coefficient method. Additionally, we implement an acceptable gap penalty for regions of 0 depth. That is, if the gap of 0 depth is small between 2 regions of similar depths, there will be a greater likelihood that they will be merged rather than a much larger one. By default, the tool utilizes the equation:

This is simply the calculation of all nonzero depths for each chromosome/reference and calculate the 68th percentile.

$$J_{threshold} = [mean_c\{Q_{68}(c)\}] + 1$$

At which point you merge a region:

$$gap_i \leq \mathrm{per}_{\mathrm{ref(c)}} \ \& \ jump_i \leq J_{threshold}$$

Signature Creation

To do this, we create a set of signature from the initial list of genomes we've used for this example:

$$S(A) = \{ min \{ h_{1(k)} \mid k \in A \}, min \{ h_{2(k)} \mid k \in A \}, \ldots, min \{ h_{n(k)} \mid k \in A \} \}$$

Where:

- $S(A)$ is the MinHash signature of set A.
- $h_i(k)$ is the hash value of k-mer k using hash function h_i

Like previously defined with the jaccard similarity, we then approximate each hash signature as such:

$$J(A, B) \approx \frac{\mid \{\mathrm{Si(A)} = \mathrm{Si(B)}\} \mid}{n}$$

While the more specific and "slower" method approaches a linear methodology (*N^2)* compute time, we've also implemented Sourmash's SBT (Bloom filter) functionality. This feature is highly scalable for many regional comparisons using the same MinHash approach. The steps to do this are:

1. Create a SBT factory the size of all of the regions present and to compare
2. Insert each Sourmash signature (**O(nk))** where n is the k-mer count
3. Query each of the signatures to the SBT factory: **(O(nk))** where n is also the k-mer count

Variability in the k-mer size and scaled values for Sourmash can alter the false positive rate. However, as mentioned before, sub-regional comparisons of alignments have shown minimal detriments to the F1 benchmark scores relative to the standard linear approaches. Ultimately, this method ensures that we have a scalable and efficient means of handling large-scale alignment results, particularly from stool or gut microbiome samples.

Generating the Confidence (S_{Comp})

Lastly, to provide a confidence value, we identify the Δ of reads + the Δ of the breadth of coverage (more in section 3 on how that is calculated).

$$\text{Confidence} = \frac{\Delta(\text{Reads}) + \Delta(\text{Breadth of Coverage})}{2}$$

Where both values contribute equality to the comparison confidence. Additionally, these reads are then filtered when considering the downstream confidence weight: B_{Comp}

C. Breadth of Coverage Weight

$$Breadth\ of\ Coverage\ (\%) = \frac{Total\ Number\ of\ Base\ Pairs\ in\ the\ Genome}{Number\ of\ Base\ Pairs\ Covered} \times 100$$

Breadth of coverage is determined post-Sourmash removal of false positives. Each alignment is provided a list of read ids to remove from upstream steps as the original alignment (BAM) file is processed. Any failed reads (those that are considered Off-targets) from the Sourmash analysis are passed over when reading in the BAM file information. This ultimately removes some breadth of coverage information for problematic regions or reads compared upstream.

Calculating Confidence Scores

The confidence score for each reference is calculated using a weighted combination of normalized features, emphasizing the uniqueness of aligned regions.

$$\mathrm{C_i} = \frac{w_1 + w_2 + w_3}{w_{\text{sum}}}$$

Where the values are bound [0,1]:

- $w_1 = S_{Comp} * (0.1)$
- $w_2 = \text{AFDS} * (0.75)$
- $w_3 = Breadth\ of\ Coverage\ weight\ * (0.15)$

Optimization Techniques

Finally, in order to identify the proper weights for each of the 3 contributors to the confidence score, we utilize a minimization function from *scipy* called ***SLSQP*** to identify the global minimum.

```
result = minimize(
        objective,
        initial_weights,
        method='SLSQP',
        bounds=bounds,
        constraints=constraints,
        options={'maxiter': max_iterations, 'disp': True}
    )
```

Where:

- Max_iterations=20
- Initial_weight=1/3 for each score
- Bounds=[0,1]

Weight Optimization

Each of the scores w_n is found across a variety of in-silico generated datasets using insilicoseq and optimizing weights based on both F1 score and the confidence disparity score ($dispc_{score}$).

$$Q^{(t)} = \alpha F_1^{(t)} + (1-\alpha) \sum_{i=1}^{N} \left(TP_i \cdot \mathrm{dispC}_{\mathrm{score}}^{(t)}(r_i) - FP_i \cdot \mathrm{dispC}_{\mathrm{score}}^{(t)}(r_i) \right)$$

where:

- $F_1^{(t)}$: F1 score evaluated for the current weights at iteration t
- $\mathrm{dispC}_{\mathrm{score}}^{(t)}(r_i)$: Confidence score for read r_i at iteration t
- α used to balance the f1 score and confidence disparity
- FP_i is the false positive indicator, 1 if true 0 if not a false positive
- TP_i is the true positive indicator, 1 if true 0 if not a true positive
- $Q^{(t)}$ is the objective function which tries to maximize/balance the f1 score and the disparity score.

This is done across multiple scenarios in order to identify the ideal weights for each scenario to apply to real-world datasets when applied.

Confidence Metric v1.0 (Old)

In order to properly identify, with confidence, the organisms present post-alignment(s), we utilize several curated pipelines/workflows using for metagenomics that are publicly available. Using benchmarks and results identified in several papers, we prioritize and weight the confidence metrics used based on their F1 Scores.

Weighted Confidence Score: The final score is a weighted sum of several factors:

$$Final\ Score = w_1\ x\ w_2\ x\ w_3\ x\ w_4\ x\ w_5$$

Where each individual w_i is a curated weight multipled by the following:

- w_1 is the MAPQ score
- w_2 is the Identity % score of Diamond post de novo assembly
- w_3 is the Disparity score
- w_4 is the Classifier Disparity between kraken2 and alignment
- w_5 is the Gini inequality coefficient score

A. Disparity Score Explanation (Weight: 30% - WIP)

The Disparity Score is a measure that assesses how skewed the aligned reads for an organism are relative to the total aligned reads in a sample. The disparity score is calculated by adjusting the proportion of reads aligned for an organism using the variance of the reads across all organisms. The idea is to account for how evenly or unevenly reads are distributed across the organisms in a sample.

Formula:

$$Disparity = Proportion\left(1 + \frac{Variance}{1 + k\ x\ Proportion}\right)$$

Where:

- **Proportion** is the total number of reads for the organism that have aligned divided by the total number of reads aligned in the sample.
- **Variance** is defined based on the number of reads aligned across all organisms, which shows indications of how spread out or concentrated the read depths are.
- **k** is a **damping factor** used to control the impact of the proportion on the variance penalty. The larger the proportion, the smaller the penalty.

Steps for Calculation:

1. **Proportion**: This simply the fraction of reads aligned to the organism relative to the total aligned reads and is also called abundance.
2. **Dynamically Dampen the Variance**: The variance of the reads across all organisms is adjusted based on the proportion of reads in total. The formula for adjusting the variance is:

$$Dampened\ Variance\ = \frac{Variance}{1\ +\ k\ x\ Proportion}$$

3. **Calculate the Disparity**: The complete disparity score is the ratio of reads multiplied by $(\ 1\ +\ \{Dampened\ Variance\}\ )$, which gives more weight towards organisms with a higher proportion of aligned reads and also less variance.

Example:

If an organism has **35%** of the total aligned reads and the variance from depths of 1000 of reads across the sample is high, the **disparity score** will be calculated as:

$$Disparity\ =\ 0.35\left(\frac{1000}{1 + 10\ x\ 0.35} + 1\right)$$

This dynamic adjustment ensures that organisms with a high amount or abundance of reads (like 35%) are not overly penalized by variance, while organisms with fewer aligned reads still feel the impact of the variance.

Key Points:

- The **damping factor k** controls how much the variance alters the disparity score. A bigger k increases the variance penalty for high-proportion organisms.
- **High-Proportion Organisms** (like those with 35% or more reads) are penalized less for variance, making their disparity more reflective of their abundance in the sample.
- **Low-Proportion Organisms** are taken into account to not unduly inflate the disparity score.

This gives a **balanced view** of how skewed the alignment is for each organism, taking into account both the **proportion** of reads and the **variance** across the entire sample. We look at variance of the number of reads across all organisms and k is a damping factor controlling the influence of the proportion on the penalty.

The normalized value of the variance is then calculated across all other organisms in the sample like so:

$$Normalized\ disparity\ = \frac{Disparity\ -\ Min.Disparity}{Max\ Disparity\ -\ Min.Disparity}$$

B. Coefficient of Variation (CV) Calculation (Weight: 5% - WIP)

The **Coefficient of Variation (CV)** is calculated as follows:

$$CV = \sigma/\mu$$

Where:

- **σ** is the standard deviation of the reads relative to all other siblings at that rank classified by kraken2.
- **μ** is the mean of the reads.

the standard deviation refers to all other sibling reads for a given taxonomic identification at the species level that has been identified with kraken2. The goal here is to determine if there is a disparate or heavily identified number of a specific organisms RELATIVE to all others in the same taxonomic rank code, in this case species (S) from K2. If Kraken2 is identifying relatively equal proportions of Escherichia (genus) species then we are less confident that that species (*E. Coli* for example) is there.

Clamping the CV Value

To make sure that the CV value remains between 0 and 1, the following clamping is applied:

$$CV_{clamped} = Max\big(0, min(CV, 1)\big)$$

This ensures that:

- If **CV** is greater than 1, it is set to 1.
- If **CV** is negative, it is set to 0.

If Kraken2 is disabled, then the pipeline will automatically reduce the exact specified weight (see default parameters for adjusting weighting) from the final confidence score. Be aware that this weight is a **WIP**.

Disparity Calculation

The **disparity** is calculated as:

$$Disparity_{cv} = 1 - CV_{clamped}$$

C. DIAMOND Identity (Weight: 30%)

$$Percent\ Identity = (Number\ of\ Matching\ Bases\ /\ Total\ Aligned\ Bases) * 100$$

D. Gini Coeff. (Weight: 30% - WIP)

The gini coefficient is a way to measure inequality in the coverage x depth of reads across a genome after alignment. It is based on the Lorenz curve, which is the cumulative proportion of the genome covered versus the cumulative proportion of the reads. In short, it is a way to identify if the total peaks or regions of depths are unequally distributed across a genome.

The score is used to figure out how evenly the reads are distributed across the genome, with values ranging from:

1: Perfect equality (all positions in the genome are equally covered by reads).
0: Maximum inequality (some positions have much higher coverage than others).

It is calculated as:

$$G = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} | X\_i - X\_j |}{2n^2X}$$

Where:

- n is the number of genomics positions across the references
- X_i is the coverage or depth at position i
- X is the mean coverage or depth, calculated as:

$$X = \frac{1}{n}\sum_{=1}^{n} X_i$$

i It is important to note that different laboratory protocols can oftentimes lead to an unavoidable disparity in coverage across a genome that can't be avoided. We are working on integrations to consider coverage differences using a positive control as a baseline for this process.

E. MapQ Score. (Weight: 5%)

This metric is simply the mean of all reads MapQ scores across an entire species/strain/subspecies, converted into a percentage.

Supplemental: Precision, Recall, and F1 Score

Precision

$$Precision = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

FP: Number of false positives.

TP: Number of true positives.

Recall

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

FN: Number of false negatives.

F1 Score

$$\text{F1Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Community Detection and Grouping

A supplemental feature explored but not introduced was tiered read removal for all multi-mapped or off-target reads. That is, we look at all of the reads across an alignment set and identify "communities" based on nt identity and read depths. While we found it useful for bam filtering, ultimately the removal of off-target hits through confidence scoring proved to be a more apt avenue for cutting out false positives while maximizing recall scores.

- **Graph Representation**: The graph $G = (V, E, W)$ is used to determine **communities** based on the similarity structure.
- **Connected Components Approach**: Communities are initially identified using **connected components**.

  Let $C = \{C_1, C_2, \dots, C_k\}$ be the set of connected components, where $C_i \subset V$ represents the nodes in component i.

Connected Components ensures that strongly connected references (such as those with manual edges) remain in the same community. This prevents any splitting that might occur due to weaker connections or the behavior of Louvain.

Tiered Read Removal Based on Disparity

Once communities are formed, reads are **reassigned** and **filtered** based on disparity:

Disparity Ratio Calculation

For each community C_i with references $\mathrm{R}_1,\ \mathrm{R}_2,\ \dots,\ \mathrm{R}_\mathrm{n}$:

1. The dominant reference R_d is the one with the **maximum read count**:

$$R_d = \arg\max_{R_k \in C_i} |R_k|$$

Where $|R_k|$ is the number of reads assigned to reference R_k.

Disparity Ratio:

For each reference $R_j \neq R_d$

$$D(R_j) = \frac{|R_d|}{|R_j|}$$

Where $D(R_j)$ is the **disparity ratio** between the dominant reference and R_j.

Tiered Removal Probability

For each reference $R_j \neq R_d$:

The **removal probability** P_{remove} is determined based on the **disparity ratio**:

$$P_{remove}(R_j) = \begin{cases} 0.9 if D(R_j) \geq 50 \\ 0.7 if 20 \leq D(R_j) < 50 \\ 0.5 if 10 \leq D(R_j) < 20 \\ 0.3 if 5 \leq D(R_j) < 10 \\ 0.1\ otherwise \end{cases}$$

Increased Removal Factor for Low Reads

If the **dominant reference** has a read count below a given threshold (T_{low}):

Increase the **removal probability** by a factor of F_{increase}:

$$P_{\text{remove}}(R_j) = \min(P_{\text{remove}}(R_j) \times F_{\text{increase}}, 1.0)$$

Where F_{increase} is the **increased removal factor**, and the probability is capped at **1.0**.

Read Removal Decision

For each read r assigned to reference Rj: The read is **removed** with probability $P_{\text{remove}}(R_j)$.

Adjusted Standard Deviation of Depth: The formula for $adjusted_{stdepth}$ combines the variance contributions of covered and missing positions, weighted by their respective counts:

$$\sigma_a = \sqrt{\frac{\sigma_c^2 \cdot C + \mu_m^2 \cdot M}{T}}$$

Where:

- σ_a: Adjusted standard deviation of depth.

- σ_c: Standard deviation of covered positions.
- C: Number of covered positions.
- μ_m: Adjusted mean depth of missing positions (treated as a variance contribution).
- M: Number of missing positions.
- T: Total number of positions $T = C + M$