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:

- $\boldsymbol{G} = \boldsymbol{0}$ Perfect equality (uniform coverage across all positions).
- $\boldsymbol{G} = \boldsymbol{1}$ 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.

1. The Lorenz Curve

Let:

- $D = \{d_1, d_2, \dots, d_n\}$ be the list of depth values, sorted in ascending order.
- n be the number of depth values.
- C_i be the cumulative sum up to the i-th depth value.

The cumulative sum is calculated as:

$$C_i = \sum_{k=1}^{i} d_k$$

We then Normalize the cumulative sums to obtain the Lorenz curve points Li

$$L_i = \frac{C_i}{C_n}$$

$C_n = \sum_{k=1}^{n} d_k$ is the total sum of the depths in the alignment.

We also include $L_0 = 0$ to show the origin of the Lorenz curve.

2. Calculate the Area Under the Lorenz Curve (A)

The area under the Lorenz curve is approximated as:

$$A = \frac{1}{n-1} \sum_{i=0}^{n-2} \frac{L_i + L_{i+1}}{2}$$

3. Gini Coefficient (G)

The Gini coefficient is finally calculated using the area under the Lorenz curve:

$$G = 1 - 2A$$

Adjusted Fair Distribution Score Calculation

1. Breadth of Coverage (B)

Let:

- n be the total genome length or the number of positions.
- NZ be the number of positions where the depth $d_i > 0$

The breadth of coverage is:

$$B = \frac{\text{NZ}}{n}$$

Apply the logarithmic transformation to emphasize higher coverage:

$$B = \log_2(B + 1)$$

Or, in simpler terms:

$$B = log_2 \left(\frac{Number\ of\ positions\ where\ d_i > 0}{n} + 1 \right)$$

2. Penalty Factor (P)

In order to minimize the loss of confidence for larger genomes, such as eukaryotes or bacteria relative to viruses, we implement an alpha (penalty) factor to reduce the impact of the gini coefficient based on a static value.

Let:

- $M = \max(D)$be the max depth in a reference
- $\mu = \frac{1}{n}\sum_{i=1}^{n} d_i$ be the average depth for a reference
- $\alpha = 2.5$ is the penalty factor and can be dynamically adjusted from the algorithm/scripts.

For example, if $\mu > 0$ the penalty would be:

$$P = \alpha \times \frac{M - \mu}{M + \mu}$$

In another case, if $\mu = 0$ it is

$$P = \alpha \times M$$

Or, in combined set of terms:

$$P = \begin{cases} \frac{M - \mu}{M + \mu,} x\ 0.5 & \frac{if\ \mu > 0}{if\ \mu = 0} \\ 0.5\ x\ M & \end{cases}$$

3. Adjust the Gini Score (GS)

The adjusted Gini score then becomes:

$$GS = (1 - G) \times (1 - P)$$

We also normalize to [0,1] as a score so it then turns into:

$$GS = \min(1, \max(0, GS))$$

4. Final Adjusted Fair Distribution Score (AFDS)

Finally, we combine the adjusted Gini score and the breadth of coverage with weights (e.g., 0.1 and 0.9):

$$\text{AFDS} = w_g 1 \times (1 - GS) + w_b \times B$$

Where w_g = weight of the Gini inequality coefficient and w_b is the weight of the breadth of coverage. These weights were manually curated based on adjusted in in-silico ground truth true/false positives.

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 **x̄** 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" between regional depths as the threshold. This threshold is then compared like so:

$$\text{jump} = |x_{\text{current}} - x_{\text{last}}|$$

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.

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:

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

Where:

- $S_i(A)$ and $S_i(B)$ are the *i* elements of the signatures $S(A)$ and $S(B)$
- n is the number of hashes in total

The similarity scores are then calculated as:

$$S(q,r) = \frac{|\,\{\mathrm{Si(q) = Si(r)}\}\,|}{n}$$

Where:

- $S(q,r)$ is the similarity score between read q and reference r.

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).

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

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.

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

Where the values are bound [0,1]:

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

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.