Python Pipeline for Extracting Barcodes & Fragments and Creating a Quantified Output Table

Overview and Key Clarifications

1. **Literal Sequences:**
    - Chosen based on the adapters used during library creation.
    - Serve as the anchors ("left" and "right" flanks) for identifying and extracting barcodes/fragments in raw sequencing reads.
2. **Sequencing Setup:**
    - **Paired-End Illumina Reads (R1 and R2) without overlap.**
    - The **barcode** is found on Read 1 (R1), while the **fragment** is on Read 2 (R2).
3. **Software Environment:**
    - The version of bbduk2 is mentioned in the README.
    - Other tools (e.g., starcode, seqkit) and Python packages are listed in env/pipeline.yml.
4. **Distance & Threshold Configurations:**
    - Hamming distances for matching literals (±2) and barcodes (±1) are **configurable** in the pipeline's config files.
    - The ratio threshold for allowing chimeric barcodes is **also configurable**—currently set to **100%** (meaning no chimeric allowance).

Step 1: Fragment Generation and Translation

Purpose
Generate all possible DNA inserts (fragments) from reference protein sequences using a sliding window approach, and record relevant metadata in a lookup table (LUT).

1. **Translation & Back-Translation**
    - **Translation:** Convert your original DNA sequences to amino acids.
    - **Back-Translation:** Convert the amino acids back to DNA using a **Homo sapiens–optimized** codon table.
    - **Output:**
        - A **reference DNA sequence file** (with human-optimized codons) used in later steps to validate fragment identity.
2. **Fragmentation**
    - **Sliding Window Approach:** Define a fragment insertion size and generate all possible fragments from the back-translated DNA.
    - **Multiple Structures:** If you have variations in fragment overhangs, the pipeline automatically accounts for these.
3. **Lookup Table (LUT) Creation**
    - Each fragment is stored in a **structured dataframe**.
    - **Metadata:**
        - Protein name of origin.
        - Fragment structure type (e.g., different overhang designs).
        - Position within the protein.
        - Peptide sequence (amino acid translation).

Output of Step 1:

- A comprehensive LUT detailing all fragments, their metadata, and references for downstream matching.

Step 2: Barcode and Fragment Extraction

This step isolates **barcodes** and **fragments** from raw Illumina **FASTQ** files using known adapter-based "literal" sequences on both ends.

1. **Defining the Literals**
    - **Left/Right Flanks:** Based on the adapters used during library creation.
    - **Configurable Hamming Distance:** Typically **2** for matching literals, to allow minor sequencing errors.
2. **Barcode Extraction (R1)**
    - Barcodes typically appear on **Read 1**.
    - **Length Variation:** ±1 base to accommodate small shifts.
    - **Literal Matching:** Hamming distance of 2 for anchors around the barcode.
3. **Fragment Extraction (R2)**
    - Fragments typically appear on **Read 2**.
    - **Exact Length Only:** No frameshift allowed; must match the expected size exactly.
    - **Literal Matching:** Again, Hamming distance of 2 for the fragment flanks.
4. **Matching to Reference Sequences**
    - Compare extracted fragments to the reference DNA (from Step 1) with **100% identity**.
    - Any fragment failing to match exactly is discarded.
    - Only done if reference exists. Else we use all extracted fragments
5. **Pairing Barcodes with Fragments**
    - Use seqkit pair to **pair** each barcode (R1) with its corresponding fragment (R2) by read ID.
    - Paired data are stored in new FASTQ files.

Output of Step 2:

- Barcode–fragment pairs that will feed into Step 3 for clustering and classification.

Step 3: Barcode Reduction and Classification

Create a comprehensive **Barcode–Fragment** table, cluster similar barcodes to correct sequencing errors, and exclude ambiguous entries.

1. **Extracting Unique Fragments & Barcodes**
    - Gather all unique barcodes and fragments.
    - This serves both as a reference for downstream steps and to visualize library diversity.
2. **Building the Full Table**
    - Merge barcode/fragments in chunks (if needed due to file size).
    - Each row links a **fragment** to a **barcode** and an entry in the LUT.
3. **Barcode Clustering (Starcode)**
    - Starcode clusters barcodes within a **Hamming distance of 1** (configurable).
    - Each cluster is represented by a **consensus** barcode, reducing duplicates introduced by sequencing errors.
4. **Replacing Barcodes**
    - Original barcodes in the table are replaced with their **cluster consensus** versions.
5. **Filtering Ambiguous Barcodes**
    - **Single-Read Barcodes:**
        - Observed only once → Mark as **ambiguous** ("Single") and excluded from further analysis.
    - **Multi-Read Barcodes:**
        - May map to exactly one fragment (**clean**) or multiple fragments (**chimeric**).
        - For chimeric barcodes, calculate the ratio $mCount/tCount$ for the most frequent fragment.
        - **Current Threshold = 100%.** Barcodes that do not exclusively map to one fragment are deemed chimeric and excluded.
6. **Generating the Final Table**
    - **library_barcodes.csv:** Contains only valid (non-ambiguous) barcode–fragment pairs.
    - Ambiguous (single or chimeric) barcodes are saved separately and excluded from further analysis.

Output of Step 3:

- A cleaned and consolidated table linking high-confidence barcodes to fragments.

Step 4: Sample Barcode Processing

Extract and process barcodes from **RNA samples** using the same general approach as in Step 2, but now applying it to experimental data.

1. **Loading Data**
    - **Library Fragments:** (Steps 1–3)
    - **Lookup Table (LUT):** Detailed metadata.
    - **Sample Metadata:** Paths, group names, etc.
2. **Barcode Extraction (R1)**
    - Same **±1 base** rule for length variation and up to **2** mismatches for the flanking literals.
    - Key read counts are logged (e.g., total reads, extracted barcodes).
3. **Barcode Clustering (Starcode)**
    - Same Hamming distance of 1 to generate consensus barcodes.
4. **Matching Barcodes with Fragments**
    - Reduced (consensus) barcodes are matched to the library fragments from Steps 1–3.
    - Peptide sequences and relevant metadata are appended.
    - Results are sorted by read frequency (RNA counts) and saved per sample.
5. **Summary Statistics**
    - For each sample, log the number of reads, barcodes, consensus clusters, matched fragments, etc.

Output of Step 4:

- Per-sample CSV files listing matched fragments, read frequencies (RNA counts), and starcode cluster barcodes.

Step 5: Data Merging (Library Fragments + Reference Positions)

Merge the **library fragment data** (from Steps 1–4) with additional **positional** information, yielding an annotated table.

1. **Loading Data**
    - **Library Fragments:** With barcode counts, modes, etc.
    - **Fragments Position File:** Start/end positions, widths, or other structural info.
    - If the positional file is missing (e.g., NNK libraries), the script only stores library fragments with their counts.
2. **Merging & Annotation**
    - Merge on **LUTnr** (lookup number) and/or **peptide**.
    - Remove unneeded columns, rename tCount → RNAcount for clarity, reorder columns.
3. **Saving Annotated Data**
    - Output CSV contains columns like:
        - **Fragment Info:** origin_seq, mode, structure, LUTnr, peptide
        - **Positional Details:** start, end, width, sequence
        - **Counts:** mCount, RNAcount

Output of Step 5:

- An annotated library fragments CSV, enriched with positional data.

Step 6: Final Dataset Processing and Normalization

Combine data from multiple samples, normalize read counts to account for sequencing depth, and optionally trim overhangs.

1. **Loading & Combining Data**
    - Load CSVs from each sample directory.
    - Rename groups for consistency (using sample metadata).
    - Integrate library fragment data for completeness.
    - Add sequence length info from a FASTA file if required.
2. **Subsetting Data**
    - Create subsets based on user-defined conditions (e.g., group membership).
    - Tag subsets and merge back into the main dataset for comparative analyses.
3. **Normalizing Read Counts**
    - Perform **group-wise normalization** to compare samples fairly despite different sequencing depths.
4. **Aggregating Identical Fragments**
    - Combine counts for identical fragments across groups.
    - Sum total reads (tCount, mCount, RNAcount, normalized counts).
    - Track unique barcodes (BC) and LUT numbers (LUTnr).
    - Optionally calculate a "barcode-adjusted count ratio" if a fragment has multiple barcodes.
5. **Trimming Fragment Overhangs**
    - If a trimming dictionary is provided, remove backbone sequences and adjust fragment boundaries accordingly.
6. **Saving the Final Output**
    - The merged dataset is saved as a CSV file, ready for **statistical analysis** or **machine learning** workflows.

Output of Step 6:

- Final annotated dataset of all found barcodes with their corresponding fragments from all samples and the library

Conclusion

This pipeline guides you from reference protein sequences to high-confidence barcode–fragment associations, through RNA sample processing, and finally to a normalized, annotated dataset. By customizing the Hamming distances, ratio thresholds, and other parameters in the config files, you can adapt the workflow to different sequencing platforms and quality requirements.