Pipeline README

This README provides instructions for running the sRNA-seq pipeline. The pipeline processes raw FASTQ files, performs quality filtering, adapter trimming, read mapping, and quantification using various bioinformatics tools. Below are the steps required to execute the pipeline successfully.

Step 1: Prepare Input Data

1. **Raw FASTQ Files:**
   - Place the raw FASTQ files into the `in` folder.
   - Ensure that the files are in gzip-compressed format (`*.fastq.gz`).
   - Name the files according to the following pattern: `<sample_name>_R1_001.fastq.gz` or `<sample_name>_R2_001.fastq.gz` respectively.
2. **Define Samples:**
   - Edit `samples.csv` with one sample name per line.
   - This file specifies the sample names used in the pipeline.

Step 2: Execute the Pipeline

Run Pipeline Script:

- Execute the `pipeline.sh` script in the terminal alternatively:
- Right-click on the 'pipeline.sh' file and select "Run as Program" from the context menu.

About `pipeline.sh`:

The `pipeline.sh` script automates the setup and execution of the sRNA-seq pipeline. It performs the following tasks:

1. **Check Dependencies:**
   - Verifies if Conda is installed. If not, installs it.
   - Checks for the required Conda environment. If not present, creates it.
2. **Create Conda Environment:**
   - Sets up a Conda environment with the following dependencies:
     - Python 3.8.0
     - trim-galore 0.6.10
     - salmon 1.4.0
     - bowtie2 2.4.5
     - pandas 1.3.3
3. **Execute Snakemake Pipeline:**
   - Runs the Snakemake pipeline within the created Conda environment.

    - Utilizes all available CPU cores for efficient processing.

Snakemake Pipeline for Read Processing, Genome Mapping, and Quantification

This pipeline is designed to process raw sequencing data through several stages, including quality control, mapping to reference genomes, and quantification of gene expression. Each step is carefully orchestrated using the Snakemake workflow management system, ensuring reproducibility and efficient parallel processing.

1. Quality Control and Adapter Trimming

- **Objective**: Remove low-quality reads and adapter sequences from raw paired-end sequencing data.
- **Tool**: Trim Galore!
- **Method**:
    - Paired-end reads are processed to remove adapter contamination and low-quality bases using `trim_galore`.
    - Quality threshold: Reads below a Phred score of 25 and lengths shorter than 8 bp are discarded.
    - FastQC is used to generate a report on the quality of trimmed reads.
- **Inputs**:
    - Paired raw read files (`sample_R1_001.fastq.gz` and `sample_R2_001.fastq.gz`)
    - Adapter/contaminants reference file (`contaminants.fasta`)
- **Outputs**:
    - Trimmed paired reads (`{sample}_R1_001_val_1.fq.gz` and `{sample}_R2_001_val_2.fq.gz`) stored in the `out/trimmed` directory.

2. Filtering Against Nucleus and Apicoplast Genomes

- **Objective**: Filter out reads that map to the nucleus and apicoplast genomes to focus on relevant sequences.
- **Tool**: Bowtie2
- **Method**:
    - The trimmed reads are mapped to a combined reference genome (nucleus and apicoplast).
    - The pipeline retains only reads that do not map to these genomes using Bowtie2's `--un-conc-gz` option.
    - Mapping mode: Local, allowing partial alignments for sensitive filtering.
- **Inputs**:
    - Trimmed paired reads (`{sample}_R1_001_val_1.fq.gz` and `{sample}_R2_001_val_2.fq.gz`)
    - Nucleus and apicoplast combined reference genome (`nucleus_apicoplast`)
- **Outputs**:
    - Filtered reads that did not map to the reference (`{sample}_filtered.1.fastq.gz` and `{sample}_filtered.2.fastq.gz`), stored in the `out/bt2_filter` directory.

3. Genome Mapping Against the Pseudo-Genome

- **Objective**: Map filtered reads to a pseudo-genome to retain the most informative sequences.

- **Tool**: Bowtie2
- **Method**:
  - The filtered reads are mapped to the pseudo-genome using `Bowtie2` in end-to-end alignment mode.
  - Multiple alignments are allowed (`-k 3`), ensuring comprehensive mapping of reads that may map to several locations.
- **Inputs**:
  - Filtered paired reads (`{sample}_filtered.1.fastq.gz` and `{sample}_filtered.2.fastq.gz`)
  - Pseudo-genome reference (`pseudo_genome`)
- **Outputs**:
  - SAM file containing the mappings (`{sample}_pseudo_genome_mapped.sam`) stored in the `out/bt2_pseudo_genome` directory.

4. Sorting and Indexing the BAM Files

- **Objective**: Sort and index the SAM files for downstream analysis and visualization.
- **Tools**: Samtools
- **Method**:
  - The SAM files are sorted by genomic coordinates and converted to BAM format.
  - An index is created for each BAM file to enable efficient visualization in genome browsers like IGV.
- **Inputs**:
  - SAM file from genome mapping (`{sample}_pseudo_genome_mapped.sam`)
- **Outputs**:
  - Sorted BAM files (`{sample}_pseudo_genome_sorted.bam`) stored in the `out/ordered` directory.
  - Index files (`{sample}_pseudo_genome_sorted.bam.bai`) stored in the same directory.

5. Read Quantification

- **Objective**: Quantify gene expression from the BAM files using feature counting.
- **Tool**: featureCounts
- **Method**:
  - Quantification is performed using `featureCounts`, which counts paired reads aligned to genes defined in the provided GTF annotation file.
  - Overlapping reads are allowed (`-O`), and multiple mappings are considered (`-M`).
  - Output is normalized by the fraction of read overlap with each feature.
- **Inputs**:
  - Sorted BAM file (`{sample}_pseudo_genome_sorted.bam`)
  - GTF annotation file for the pseudo-genome (`RNA_index.gtf`)
- **Outputs**:
  - Quantified read counts for each sample (`{sample}.txt`) stored in the `out/quantified` directory.

Pipeline Overview

- **Conda Environment**: The pipeline operates within a conda environment, ensuring all required dependencies (e.g., `trim_galore`, `bowtie2`, `samtools`, `featureCounts`) are installed and reproducible.
- **Configuration**: The pipeline is configurable via the `config.yaml` file, which specifies paths to input files, references, and parameters.
- **Input**: Paired-end sequencing data in FASTQ format along with reference files for contaminants, pseudo-genome, and gene annotation.
- **Output**: BAM files suitable for visualization, quantified gene expression files, and intermediary files for quality control and mapping. """

Additional Notes:

- Ensure that the input files, sample names, and directory structure are correctly set up before running the pipeline.
- Check the console output for any errors or warnings during pipeline execution.
- For detailed information on each step and tool parameters, refer to the documentation provided with each tool or consult the Snakemake workflow file.