|  |  |
| --- | --- |
| **Scripting** | |
| **R** | Write an R script that performs differential expression analysis on RNA-Seq data. The script should:   1. Load a CSV file with columns: Gene ID, Sample 1, Sample 2, ..., Sample N. 2. Normalize the read counts using the DESeq2 package. 3. Perform differential expression analysis between two conditions. 4. Save the analysis results in a CSV file with columns: Gene ID, log2 Fold Change, p-value, Adjusted p-value. |
| **Python** | Write a python script that calculates the GC content of DNA sequences. The script should   1. Load a FASTA file containing DNA sequences. 2. Calculate the GC content for each sequence. 3. 3. Output the results in a CSV file with columns: Sequence ID, GC Content. |
| **Snakemake** | Create a simple Snakefile for a bioinformatics workflow using Snakemake and conda, including:   * Trimmomatic: Trim and clean raw sequencing data. * FastQC: Perform quality control on trimmed data.   Requirements:   * Trimmomatic Rules: Process all input files in the raw\_data directory. * FastQC Rules: Run only on trimmed data.   Conda Configuration:   * Include conda configurations for each tool. * Use a single environment.yml file for the entire workflow. * Snakemake 8.25.5 is already installed. Do not include this in the yml file   Input Data:   * Unzipped FASTQ files in {input\_directory}/data.   Deliverables:   * Snakefile with specified rules and configurations. * - environment.yml file with dependencies. |
| **Code Conversion** | |
| **Nextflow** | Persona: You are a bioinformatician with no experience running nextflow workflows. You need to convert your code from a scripting language to an nf-core workflow.  Task: Convert this code to an nf-core workflow.  Details:   * Source Code: The code is written in BASH * Target Workflow: The code needs to be converted into an nf-core workflow. * Input Format: The nextflow script should accept input files in a samplesheet format.   Additional Requirements:   * Ensure that the nf-core workflow adheres to best practices. * Include necessary modules and processes to replicate the functionality of the original script. * Provide clear comments and documentation within the script for ease of understanding and future modifications. * Ensure that the nextflow\_schema.json file includes the script name and a description of the workflow.   Desired Output:   * A nf-core style set of scripts that replicates the above functionality. The necessary files include the main.nf file, a nextflow.config file, a nextflow\_schema.json, and a modules.config file. * The script should accept input files in a samplesheet format. * Please use a publishDir directive anytime a directory must be created * The script should be able to run in a docker, singularity or conda environment.   Script to convert:  bacass.sh  ```  #!/bin/bash  # Directory containing the input FASTQ files  INPUT\_DIR="/home/ramsivakumar/nextflow\_conversion/fastq"  OUTPUT\_DIR="/home/ramsivakumar/nextflow\_conversion/test\_out\_bash"  # Create output directories for the tools  mkdir -p "$OUTPUT\_DIR/fastp\_output" "$OUTPUT\_DIR/fastqc\_output" "$OUTPUT\_DIR/unicycler\_output"  # Loop over the files in the input directory  fastq\_file\_r1="$INPUT\_DIR/${sample\_name}\_R1\_001.fastq.gz"  fastq\_file\_r2="$INPUT\_DIR/${sample\_name}\_R2\_001.fastq.gz"  # Check if the R2 file exists  if [[ ! -f "$fastq\_file\_r2" ]]; then  echo "Warning: Corresponding R2 file for $fastq\_file\_r1 not found. Skipping this pair."  continue  fi  # Step 1: Run fastp  fastp\_output\_r1="$OUTPUT\_DIR/fastp\_output/${sample\_name}\_R1.fastp.fastq"  fastp\_output\_r2="$OUTPUT\_DIR/fastp\_output/${sample\_name}\_R2.fastp.fastq"  fastp -i "$fastq\_file\_r1" -I "$fastq\_file\_r2" -o "$fastp\_output\_r1" -O "$fastp\_output\_r2"  # Step 2: Run fastqc on the fastp output  fastqc\_output\_dir="$OUTPUT\_DIR/fastqc\_output/${sample\_name}\_fastqc"  mkdir -p "$fastqc\_output\_dir"  fastqc "$fastp\_output\_r1" "$fastp\_output\_r2" -o "$fastqc\_output\_dir"  # Step 3: Run multiqc  multiqc\_output\_dir="$OUTPUT\_DIR/multiqc\_output"  mkdir -p "$multiqc\_output\_dir"  multiqc "$fastqc\_output\_dir" -o "$multiqc\_output\_dir"  # Step 4: Run unicycler on the fastp output  unicycler\_output\_dir="$OUTPUT\_DIR/unicycler\_output/${sample\_name}\_unicycler"  mkdir -p "$unicycler\_output\_dir"  unicycler -1 "$fastp\_output\_r1" -2 "$fastp\_output\_r2" -o "$unicycler\_output\_dir"  echo "Finished processing $sample\_name."  done  echo "All files processed."  ``` |
| **Code** **Optimization** | |
| **BASH** | Optimize this script to improve readability and efficiency:  ```  #!/bin/bash  # Directory containing the input FASTQ files  INPUT\_DIR="/mnt/c/Users/ramsivakumar/chatbot\_testing/data"  OUTPUT\_DIR="/mnt/c/Users/ramsivakumar/chatbot\_testing/bacass\_optim\_pre"  # Create output directories for the tools  mkdir -p "$OUTPUT\_DIR/fastp\_output" "$OUTPUT\_DIR/fastqc\_output" "$OUTPUT\_DIR/unicycler\_output"  # Loop over the files in the input directory  for fastq\_file\_r1 in "$INPUT\_DIR"/\*\_1.fastq.gz; do  sample\_name=$(basename "$fastq\_file\_r1" "\_1.fastq.gz")  fastq\_file\_r2="$INPUT\_DIR/${sample\_name}\_2.fastq.gz"  # Check if the R2 file exists  if [[ ! -f "$fastq\_file\_r2" ]]; then  echo "Warning: Corresponding R2 file for $fastq\_file\_r1 not found. Skipping this pair."  continue  fi  # Step 1: Run fastp  fastp\_output\_r1="$OUTPUT\_DIR/fastp\_output/${sample\_name}\_R1.fastp.fastq"  fastp\_output\_r2="$OUTPUT\_DIR/fastp\_output/${sample\_name}\_R2.fastp.fastq"  fastp -i "$fastq\_file\_r1" -I "$fastq\_file\_r2" -o "$fastp\_output\_r1" -O "$fastp\_output\_r2"  # Step 2: Run fastqc on the fastp output  fastqc\_output\_dir="$OUTPUT\_DIR/fastqc\_output/${sample\_name}\_fastqc"  mkdir -p "$fastqc\_output\_dir"  fastqc "$fastp\_output\_r1" "$fastp\_output\_r2" -o "$fastqc\_output\_dir"  # Step 3: Run multiqc  multiqc\_output\_dir="$OUTPUT\_DIR/multiqc\_output"  mkdir -p "$multiqc\_output\_dir"  multiqc "$fastqc\_output\_dir" -o "$multiqc\_output\_dir"  # Step 4: Run unicycler on the fastp output  unicycler\_output\_dir="$OUTPUT\_DIR/unicycler\_output/${sample\_name}\_unicycler"  mkdir -p "$unicycler\_output\_dir"  unicycler -1 "$fastp\_output\_r1" -2 "$fastp\_output\_r2" -o "$unicycler\_output\_dir"  echo "Finished processing $sample\_name."  done  echo "All files processed."  ``` |
| **Error** **Debugging** | |
| **WDL** | Fix the errors in this WDL script:  ```  workflow samtools\_sort\_index\_workflow {  input {  File input\_bam  }  call samtools\_sort {  input:  bam\_file = input\_bam  }  call samtools\_index {  input:  sorted\_bam = samtools\_sort.sorted\_bam  }  output {  File sorted\_bam = samtools\_sort.sorted\_bam  File bam\_index = samtools\_index.bam\_index  }  }  task samtools\_sort {  input {  File bam\_file  }  command {  samtools sort ${bam\_file} -o sorted.bam  }  output {  File sorted\_bam = "sorted.bam"  }  runtime {  docker: "biocontainers/samtools:v1.9-4-deb\_cv1"  memory: "4G"  cpu: 2  }  }  task samtools\_index {  input {  File sorted\_bam  }  command {  samtools index ${sorted\_bam}  }  output {  File bam\_index = "sorted.bam.bai"  }  runtime {  docker: "biocontainers/samtools:v1.9-4-deb\_cv1"  memory: "2G"  cpu: 1  }  }  ```  Corrected script:  ```  workflow samtools\_sort\_index\_workflow {  input {  File input\_bam  }  call samtools\_sort {  input:  bam\_file = input\_bam,  }  call samtools\_index {  input:  sorted\_bam = samtools\_sort.sorted\_bam  }  output {  File sorted\_bam = samtools\_sort.sorted\_bam  File bam\_index = samtools\_index.bam\_index  }  }  task samtools\_sort {  input {  File bam\_file  }  command {  samtools sort ${bam\_file} -o sorted.bam  }  output {  File sorted\_bam = "sorted.bam"  }  runtime {  docker: "biocontainers/samtools:v1.9-4-deb\_cv1"  memory: "4G"  cpu: 2  }  }  task samtools\_index {  input {  File sorted\_bam  }  command {  samtools index ${sorted\_bam}  }  output {  File bam\_index = "sorted.bam.bai"  }  runtime {  docker: "biocontainers/samtools:v1.9-4-deb\_cv1"  memory: "2G"  cpu: 1  }  }  ```  Explanation of Corrections  Workflow Call Input Mapping:   * Added a comma at the end of the input mapping line in the call samtools\_sort block. * Ensured that the call samtools\_index block correctly references the output from the samtools\_sort task.   Output Declaration in Workflow:   * Correctly referenced the outputs from the samtools\_sort and samtools\_index tasks in the output block of the workflow. |
| **Knowledge** **Test** | |
| **GCP** | * AI: How can I create and test a RAG system on Google Cloud? * Data and Analytics: What are the key features of Google Vertex AI for data science workflows? * Compute Services: How do you configure Auto Scaling for compute engine in Google Cloud? |
| **AWS** | * AI: How can I create and test a RAG system on AWS? * Data and Analytics: What are the key features of Amazon SageMaker for data science workflows? * Compute Services: How do you configure Auto Scaling for EC2 instances in AWS? |
| **Azure** | * AI: How can I create and test a RAG system on Azure? * Data and Analytics: What are the key features of Azure Machine Learning for data science workflows? * Compute Services: How do you configure Auto Scaling for virtual machines instances in Azure? |
| **Workflow Managers** | * Snakemake: How do you define a rule in Snakemake, and what are the essential components of a rule? * - Nextflow: What is the purpose of a main.nf file in a nextflow workflow? |

Data: <https://github.com/STRIDES/NIHCloudLabAWS/tree/drafts/docs/chatbot_comparison/data>