ASC Student Supercomputer Challenge 2025

Preliminary Round Announcement

Dear ASC25 Teams:

Welcome to the 2025 edition of ASC Student Supercomputer Challenge ASC25!

The ASC Student Supercomputer Challenge (ASC25) marks its 13th year as the world's largest supercomputing hackathon, continuing its mission to cultivate young talent and drive innovation in Supercomputing and AI. Since its launch at SC24 on November 21, 2024, ASC25 has attracted immense interest, with hundreds of teams registered to compete. The competition progresses to its Preliminary phase on January 6, 2025, promising another exciting chapter of innovation and collaboration.

Preliminary Round: Teams Gear Up for Submission.

In the Preliminary round of ASC25, teams are encouraged to put forth their best efforts to complete the assigned tasks and submit comprehensive proposal documentation. Submissions should include detailed cluster design, source code optimization approaches, and output files. The ASC25 evaluation committee will rigorously review all proposals in English, ensuring a fair and thorough assessment.

Submission Guidelines.

Important Deadline. All participating teams must submit the required materials by 24:00, February 21, 2025 (UTC/GMT +8:00).

a) **Proposal Submission**. Name the proposal using the format: [University/College Name]_[Contact Person Name] (e.g., ABC_University_John_Doe). Save the proposal as a single PDF file. Upload the PDF file at the official ASC25 website: https://www.asc-events.net/StudentChallenge/ASC25/PreliminarySubmission.php

b) **Additional Materials**. All the additional information should be compressed into one file using the same naming convention: ABC_University_John_Doe. The compressed file should include at least four folders, per the requirements detailed in Appendix A. Please upload the compressed file to the FTP server. The FTP server address will be provided via email at a later date.

- Output files of HPL
- Output files of HPCG
- Required files of AlphaFold3 inference challenge
- Required files of RNA m^5C challenge

c) **Submission Confirmation and Support**. Ensure all submissions are completed and submitted on time, adhering to the guidelines, or there will be no grade awarded. A

confirmation email will be sent shortly after all required materials have been successfully received. For any further inquiries or assistance, please contact:

- Technical Support: techsupport@asc-events.org
- General Information: info@asc-events.org
- Press: media@asc-events.org

The ASC25 Committee extends its best wishes to all teams as you embark on your ASC25 journey.

Appendix A:

Proposal Requirements.

I. Introduction to the university's activities in supercomputing (5 points)

1. Supercomputing-related hardware and software platforms.
2. Supercomputing-related courses, trainings, and interest groups.
3. Supercomputing-related research and applications.
4. A brief description of the key achievements on supercomputing research, no more than 2 items.

II. Team introduction (5 points)

1. Brief description of the team setup.
2. Introduction and the photo of each member, including group photos of the team.
3. Team's motto or catch-phrase.

III. Technical proposal requirements (90 points)

1. Design of HPC system (15 points)

Participating teams are required to submit a theoretical design of an HPC cluster, rather than constructing a physical cluster. The designed cluster must meet the following requirements:

a) The system should be designed for optimal computing performance. The cluster must consist of at least three compute nodes, with a per-node power limit of 2000 W and a total power limit of 4000 W for all cluster components.

b) Specify the system's software and hardware configurations, as well as its interconnection. Describe the power consumption, evaluate the performance, and analyze the advantages and disadvantages of the proposed architecture.

Note: The hardware components listed in the table below are provided for reference only. They are based on a dual processor server, which supports up to 2 GPUs.

Item	Name	Configuration
Server	Dual Processor Server	CPU: Intel® Xeon® 6760P Processor * 2 Memory: 32GB * 16, DDR5, 6400 MT/s Hard disk: 480GB SSD SATA * 1
HCA card	NDR200	InfiniBand NVIDIA ConnectX®-7 NDR200
Switch	GbE switch	10/100/1000 MB/s，24 ports Ethernet switch
	NDR-IB switch	NVIDIA Quantum (TM)-2 NDR InfiniBand Switch, 64-ports NDR, 32 OSFP ports, unmanaged, P2C airflow (forward)
Cable	Gigabit CAT6 cables	CAT6 copper cable, blue, 3 m
	InfiniBand cable	InfiniBand NDR copper cable, OSFP port, compatible with the InfiniBand switch in use.

Note: The hardware configuration in the ASC25 competition finals may be different from the table above.

2. HPL and HPCG Benchmarks (15 points)

The proposal should include descriptions of the software environment (operating system, complier, math library, MPI software, software version, etc.), the performance optimization and testing methods, performance measurement, problem and solution analysis, etc. In-depth analysis on HPL, HPCG algorithms and the respective source codes would be a plus.

Download the HPL software at: http://www.netlib.org/benchmark/hpl/.

Download the HPCG software at: https://github.com/hpcg-benchmark/hpcg

It is recommended to run verification and optimization of HPL and HPCG benchmarks on x86 CPU and Data Center GPU platforms. If other hardware platforms are used, you are welcomed to submit the related analysis and results that demonstrate adequate performance.

3. Optimization for AlphaFold3 Inference (30 points)

Task Description

Protein structure prediction is a fundamental and challenging task that has remained a significant hurdle in biology for half a century. The revolutionary AI model AlphaFold successfully solved this decades-old scientific challenge with remarkable performance, earning it half of the 2024 Nobel Prize in Chemistry. The latest iteration, AlphaFold3, equipped with novel features, can accurately predict the structures of complexes involving ligands, proteins, and nucleic acids, potentially revolutionizing drug discovery, disease treatment, and our understanding of fundamental biological processes.

AlphaFold3 employs a diffusion-based architecture and begins with an input of either an amino acid sequence or sequences of multiple biomolecules. The process consists of two stages: the data pipeline and model inference. The data pipeline, running on CPUs, constructs MSAs for protein and RNA entities using Jackhmmer/Nhmmer searches across genetic databases. The model inference stage, operating on GPUs, utilizes the MSAs, templates, and original sequences, passing them through the Pairformer and diffusion modules to generate the predicted structures.

In the Preliminary round, the focus is solely on the model inference stage. The committee provides twelve samples of single protein sequences along with pre-generated MSAs and templates as task inputs. There is no need to execute the data pipeline stage. The objective of the preliminary task is **to minimize inference time**, focusing on the following two aspects:

(1) Optimize the GPU inference process and minimize the inference time. (10 points)

(2) Migrate the inference codebase from GPU to CPU architecture and optimize it to minimize the corresponding CPU inference time. (20 points)

For instance, the relevant inference time is 53.76 seconds, as shown on the screen capture, after running the inference stage.

```
Featurising 2PV7 with rng_seed 1 took 5.63 seconds.
Featurising data for seeds (1,) took 8.54 seconds.
Running model inference for seed 1...
Running model inference for seed 1 took 53.76 seconds.
Extracting output structures (one per sample) for seed 1...
Extracting output structures (one per sample) for seed 1 took 0.42 seconds.
Running model inference and extracting output structures for seed 1 took 54.19 seconds.
Running model inference and extracting output structures for seeds (1,) took 54.19 seconds.
Writing outputs for 2PV7 for seed(s) (1,)...
Done processing fold input 2PV7.
Done processing 1 fold inputs.
```

Once AlphaFold3 is installed, one can run the application using the following command to skip the data pipeline:

```
cd alphafold
python run_alphafold.py --json_path=json_file_path --model_dir=model_parameter_path --norun_data_pipeline --output_dir=output_DIR
```

Upon completion of the inference process, an output directory aligned with the ‘name’ field in the input file is expected to be generated. Below is an example of an AlphaFold3 output directory listing for a job named ‘hello_fold’, which was run with one seed.

```
hello_fold/
├── seed-1_sample-0/
│   ├── confidences.json
│   ├── model.cif
```

```
│   └── summary_confidences.json
├── seed-1_sample-1/
│   ├── confidences.json
│   ├── model.cif
│   └── summary_confidences.json
├── seed-1_sample-2/
│   ├── confidences.json
│   ├── model.cif
│   └── summary_confidences.json
├── seed-1_sample-3/
│   ├── confidences.json
│   ├── model.cif
│   └── summary_confidences.json
├── seed-1_sample-4/
│   ├── confidences.json
│   ├── model.cif
│   └── summary_confidences.json
├── TERMS_OF_USE.md
├── hello_fold_confidences.json
├── hello_fold_data.json
├── hello_fold_model.cif
├── hello_fold_summary_confidences.json
└── ranking_scores.csv
```

Note

(1) AlphaFold3 can be accessed through the GitHub repository https://github.com/google-deepmind/alphafold3. Note that ASC Committee utilized version 3.0.0 for testing.

(2) The model parameters of AlphaFold3 can be obtained upon request by completing the form: https://forms.gle/svvpY4u2jsHEwWYS6.

(3) In our case the MSA information is already incorporated within the input files, multiple genetic (sequence) protein and RNA databases are not needed. The twelve input files can be downloaded from the ASC repository: https://github.com/ASC-Competition.

Result Submission

After optimization, run all twelve input cases on the GPU and CPU separately, and upload the corresponding results to the ASC25 official FTP server. Note that the size of the output files will exceed 1 GB. Therefore, compress the files and name the package AlphaFold3.tar.gz before uploading. The directory structure is outlined below:

```
AlphaFold3
    |___GPU-optimization          (script or code files used in the GPU inference process)
    |___GPU-results
            |___case_name_1       (Output directory generated during the inference. Use
```

the default directory name, which corresponds to the ‘name’ field within the input file. The directory

structure is resembles to the ‘hello_fold’ example mentioned above and the specific details of this structure are omitted here for simplicity. The same applies hereinafter for the remaining cases.)

```
|___case_name_2
|___……
|___case_name_12
|___case_name_1.log      (Screen output during inference procedure)
|___case_name_2.log
|___……
|___case_name_12.log
|___SUMMARY_time.out   (A summary file that includes the running time before and after the optimization for all cases.)
|___CPU-optimization          (script or code files used in the CPU inference process)
|___CPU-results
    |___case_name_1
    |___case_name_2
    |___……
    |___case_name_12
    |___case_name_1.log
    |___case_name_2.log
    |___……
    |___case_name_12.log
    |___SUMMARY_time.out
```

Evaluation

During the evaluation process, the ASC25 Committee will primarily focus on improvements in inference performance on both CPU and GPU. Additionally, the detailed optimization strategies and the underlying principles outlined in the proposal will also serve as a basis for scoring. Below are a few important points to note:

(1) BF16 is the default precision, and all precision below 16-bit is not allowed.

(2) Do not modify the AlphaFold3 model parameters.

(3) Do not modify the model_config.json file, including but not limited to the recycling number and diffusion number.

(4) Do not modify the provided input JSON files, including modelSeeds, sequences, and MSA, etc.

(5) Please maintain the directory structure and the resulting files unchanged. Do not alter the output files of the original AlphaFold3, including but not limited to the structure files and confidence files.

(6) Both the inference speed and the accuracy of structure prediction hold significance in the evaluation. Evidently unphysical structures will result in a deduction of points.

(7) The submitted inference log files must contain details such as the prediction start time, time consumption for each stage (including inference, extracting output structures, etc.),

and other essential elements of the inference process.

(8) Participants are required to provide comprehensive details about the model inference process, machine specifications, environment setup, optimization methods used, and performance comparison in the proposal. These details will serve as the primary basis for scoring by the ASC25 Committee.

(9) Participants must submit all the required files as specified above; otherwise, the score for this part will be set to 0.

4. RNA m^5C Modification Site Detection and Performance Optimization Challenge (30 points)

Task Description

RNA molecules undergo various chemical modifications that significantly influence gene expression regulation, post-transcriptional processes, and protein translation. To date, over 170 types of modifications have been identified in RNAs. Among these, 5-methylcytosine (m^5C) is a key modification, widely distributed across diverse RNA species and playing a critical role in gene expression and its regulation.

With the advent of high-throughput sequencing (HTS) technologies, several methods—such as RNA-Bis-seq and UBS-seq—have been developed to detect m^5C at single-base resolution. However, most of these methods rely on the base-conversion signal from cytosine (C) to thymine (T), which inherently increase the risk of generating high false-positive rates. Balancing the accuracy and reliability of m^5C detection while minimizing false positives remains a key challenge in current m^5C research.

This task involves implementing a workflow that systematically incorporates a suite of bioinformatics software packages specifically designed to detect RNA m^5C in HTS data. The objective is to refine the provided analysis pipelines to enhance the accuracy and reliability of m^5C site detection while minimizing runtime. Evaluation criteria for this task will emphasize the precision of m^5C site identification, control of false positives, and computational efficiency.

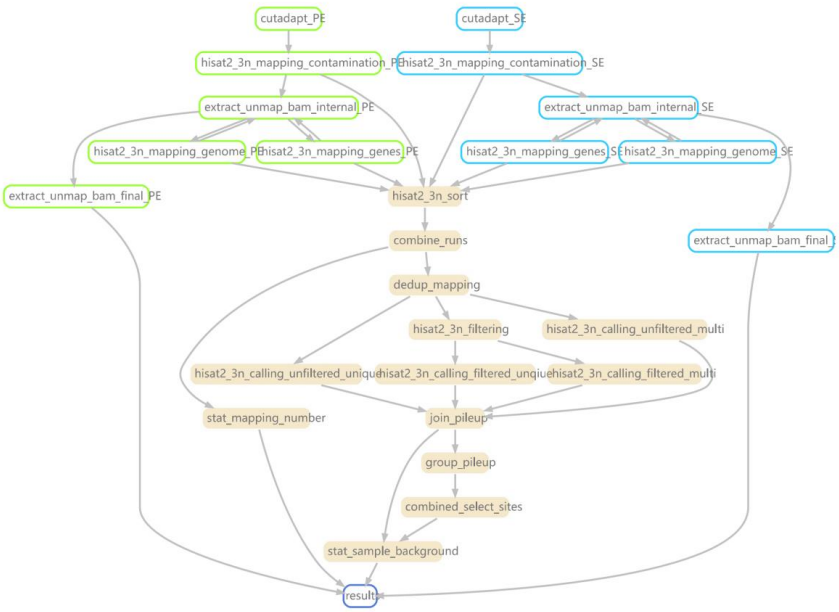
It is important to note that, unlike workflows that rely on a single tool, this task integrates multiple software components. Starting from raw sequencing data (FASTQ files), the pipeline performs adapter trimming, low-quality sequence removal, rRNA/tRNA filtering, genome alignment, m^5C site detection, and stringent filtering. The final output is a high-confidence list of m^5C candidate sites with both high precision and low false-positive rates.

The reference paper: Dai, Q., Ye, C., Irkliyenko, I. et al. Ultrafast bisulfite sequencing detection of 5-methylcytosine in DNA and RNA. Nat Biotechnol 42, 1559–1570 (2024). https://doi.org/10.1038/s41587-023-02034-w

The reference code: https://github.com/y9c/m5C-UBSseq

In addition, the main analysis workflows are provided under the docs directory. Also, the “Snakefile” is very useful.

Main analysis workflow



Datasets

Link to raw input data: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE225614

GSM7051146 WT HeLa polyA+ RNA treated with ultrafast BS, replicate 1
GSM7051147 WT HeLa polyA+ RNA treated with ultrafast BS, replicate 2
GSM7051148 WT HeLa polyA+ RNA treated with ultrafast BS, replicate 3

It is recommended to use the **SRA Toolkit** to download and extract datasets.

Guidelines for the procedures

(1) Reference Index Construction: Build separate indices for rRNA, tRNA and the genome after performing a C→T base replacement in the reference, ensuring compatibility with subsequent C→T-based detection steps.

Input: rRNA, tRNA and genome reference sequence

Output: rRNA index, tRNA index, genome index, plus corresponding FAI and SAF files for future alignment and variant calling.

(2) Data Cleaning: Remove adapter sequences, trim low-quality bases, and process poly(A) tails, resulting in clean reads for downstream analysis.

Input: Raw FASTQ files

Output: Cleaned FASTQ files, a record of reads removed for being too short, and any untrimmed reads.

(3) rRNA Alignment: Align the cleaned reads to the rRNA reference to remove rRNA contamination.

Input: Cleaned FASTQ files.

Output: Reads mapped to rRNA, unmapped reads, alignment statistics, unmapped FASTQ.

(4) tRNA Alignment: Filter out tRNA contamination by aligning unmapped reads from the rRNA alignment step to a tRNA reference.

Input: FASTQ reads unmapped to rRNA.

Output: Reads mapped to tRNA, unmapped reads, alignment statistics, unmapped FASTQ.

(5) Genome Alignment: Align the rRNA- and tRNA-filtered reads to the genome reference (built with C→T indexes).

Input: FASTQ reads unmapped to rRNA/tRNA.

Output: Genome-aligned reads, unmapped reads, alignment statistics.

(6) Sorting BAM Files: Sort the BAM files for easier downstream processing

Input: Aligned BAM files.

Output: Sorted BAM files.

(7) Merging, Statistics, and Deduplication: Merge sorted alignments, calculate alignment statistics, and remove duplicate reads that might inflate false-positive signals.

Input: Sorted BAM files

Output: (i) Merged alignment statistics (TSV), (ii) deduplicated BAM files, and corresponding log files.

After deduplication, index the final BAM (.bam.bai).

(8) Site Calling and Filtering: Identify candidate C→T conversion sites. Use separate "unfiltered" and "filtered" conditions to categorize unique vs. multi-mapped reads.

Input: Deduplicated BAM files

Output: Four TSV files containing statistics for each condition: unfiltered_uniq, unfiltered_multi, filtered_uniq, filtered_multi

join_pileup: For each sample, consolidate base count information (converted vs. unconverted) from the four conditions into a single file.

Script: join_pileup.py

Input: The four TSV outputs (unfiltered_uniq, unfiltered_multi, filtered_uniq, filtered_multi)

Output: One .arrow file, merging the base count data based on ref, pos, and strand.

group_pileup: Merge data across multiple replicates (all samples in one group), compute coverage and key metrics for each site, and identify preliminary m^5C candidates at the group level.

Script: group_pileup.py

Input: .arrow files from multiple samples in the same group

Output: A single .arrow containing combined coverage and ratio metrics.

Calculated Metrics:

u: Sum of unconverted base counts under filtered_uniq across all samples

d: Total coverage (converted + unconverted) under filtered_uniq across all samples

ur: Unconverted ratio, u / d

mr: Multiple-mapping ratio, unfilter_multi / (unfilter_multi + unfilter_uniq)

cr: "Cluster ratio," proportion of reads lost when moving from unfiltered to filtered conditions.

$$1 - \frac{\text{filtered\_multi} + \text{filtered\_uniq}}{\text{unfiltered\_multi} + \text{unfiltered\_uniq}}$$

combined_select_sites: Apply preliminary thresholds to identify candidate m^5C sites.

Script: select_sites.py

Input: Per-group .arrow files

Output: A .tsv file listing candidate m^5C sites (only retaining essential columns such as ref, pos, strand).

Filtering Criteria (for each site):

d >= TOTAL_DEPTH (default ≥ 20)

u >= TOTAL_SUPPORT (default ≥ 3)

ur >= AVERAGE_UNC_RATIO (default ≥ 0.02)

cr < AVERAGE_CLU_RATIO (default < 0.5)

mr < AVERAGE_MUL_RATIO (default < 0.2)

stat_sample_background: Perform background estimation and statistical testing (binomial test) for each sample to confirm final m^5C candidates.

Script: filter_sites.py

Input:

The .arrow files from join_pileup for each sample

The .tsv candidate sites from combined_select_sites

Output:

A file storing the global background methylation ratio (bg_ratio) for each sample.

A file listing each site's final detection result, including ref, pos, strand, u, d, ur, pval, and a boolean passed column.

Logic:

Background Estimation: Remove all candidate sites from the dataset and calculate the average unconverted ratio (ur) from the remaining positions to get bg_ratio.

Significance Testing: Perform a binomial test for each candidate site.

If pval < 0.001 and (u >= 2), (d >= 10), (ur > 0.02), the site is called m^5C in that sample.

$$p\_val = \text{binomtest}(\text{successes} = u, \text{trials} = d, p = bg\_ratio)$$

(9) Combine Across Replicates: Retain only sites with p-value < 10^{-6} across all three replicates as final m^5C sites.

Processing Software Guidance and Parameters

The following are the recommended process and parameter guidelines, which can be adjusted appropriately according to the actual amount of data and computing resources:

(1) Construction of the reference index:

Software: hisat3n-build, samtools

Recommended parameters (hisat3n-build): -p 12 --base-change C,T

(2) Data cleaning

Software: cutseq

Recommended parameters: -t 20 -A INLINE -m 20 --trim-polyA --ensure-inline-barcode

(3) rRNA, tRNA filtering and genome alignment

Software: hisat3n, samtools

Parameters for rRNA and tRNA alignment (hisat3n): --base-change C,T --mp 8,2 --no-spliced-alignment --directional-mapping

Parameters for genome alignment (hisat3n): --base-change C,T --pen-noncansplice 20 --mp 4,1 --directional-mapping

(4) Sorting and deduplication

Software: samtools sort, java + umicollapse.jar

Recommended parameters (samtools): -@ 20 -m 3G --write-index

Recommended parameters (umicollapse): bam -t 2 -T 20 --data naive --merge avgqual --two-pass

(5) Site calling and filtering

Software: samtools view, hisat3n-table, bgzip

Main parameters (samtools): -e "rlen<100000"

Main parameters (hisat3n-table): -p N, -u/-m, --alignments, --ref..., --base-change C,T

Main parameters (samtools view): -@ 10, -e "[XM]20 <= (qlen-sclen) && [Zf] <= 3 && 3[Zf] <= [Zf]+[Yf]"

(6) The following scripts can be found on GitHub in details.

Script Name	Function Description	Dependent Packages	Main Methods
join_pileup.py	Integrate the statistical information of the same sample under different conditions.	argparse, polars	pl.read_csv, pl.join, pl.fill_null
group_pileup.py	Integrate the statistical information of multiple samples in the same group, calculate key indicators.	argparse, polars	pl.read_ipc, pl.join, pl.sum_horizontal
select_sites.py	Conduct a preliminary screening on the group statistical results, filter out the sites that do not meet the thresholds, and output the candidate sites.	argparse, polars	pl.read_ipc, pl.filter, pl.unique
Filter_sites.py	Calculate the proportion of the background that has not been converted, conduct a significance test, and screen the final m^5C candidate sites.	argparse, polars, scipy	pl.join, pl.with_columns, binomtest

UBS-seq reference code: https://github.com/y9c/m5C-UBSseq

Results Submission

(1) Workflow description file

Includes intermediate file names (e.g., alignment rates and QC summaries).

Code Documentation: If any script or program was modified, provide a file explaining the changes and the rationale.

In addition, the running time of each step in the workflow should also be provided, the "time" command can be used to obtain the execution time of each step.

(2) m^5C sites file

Filtered TSV files of **three** datasets, for example, SRR23538290.filtered.tsv, SRR23538291.filtered.tsv and SRR23538292.filtered.tsv.

(3) Software Packaging

Packaging the entire workflow into a single software tool or container (e.g., using conda to encapsulate all environment configurations and software settings) would streamline the installation and execution process, reduce environment-related errors, and promote reproducibility.

(4) Record and submit the time elapsed from the start of "**cutseq**" to the end of the workflow by means of screenshots.

(5) Please compress all the required files and name the package as RNA.tar.gz.

Result Evaluation

Precision: Proportion of correctly detected sites (true positives, TP) among all detected sites.

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

Where TP are the sites matching the standard answer and FP are those not in the standard.

Reference Shell Code:

Precision=$(awk 'NR==FNR {a[$1,$2,$3]=1; next} ($1,$2,$3) in a' "true.tsv" "detected.tsv" | wc -l | awk -v total=$(wc -l < "$detected_file") '{printf "%.2f", ($1/total)*100}')

Correlation: Calculating the Pearson correlation of unconverted ratios (ur) for the overlap sites between the participant's detected sites and the true sites.

$$r = \frac{\sum_{k=1}^{n}\left(ur_{\text{true},k} - \overline{ur_{\text{true}}}\right)\left(ur_{\text{detected},k} - \overline{ur_{\text{detected}}}\right)}{\sqrt{\sum_{k=1}^{n}\left(ur_{\text{true},k} - \overline{ur_{\text{true}}}\right)^2}\sqrt{\sum_{k=1}^{n}\left(ur_{\text{detected},k} - \overline{ur_{\text{detected}}}\right)^2}}$$

Reference Shell Code:

```
paste true_ur.txt detected_ur.txt | awk '{
    sumXY += $1 * $2;
    sumX += $1;
    sumY += $2;
    sumX2 += $1 * $1;
    sumY2 += $2 * $2;
    n++;
} END {
    print (n * sumXY - sumX * sumY) / (sqrt(n * sumX2 - sumX^2) * sqrt(n * sumY2 - sumY^2))
}'
```

Accuracy: Must meet specified thresholds of precision (>=95%) and correlation (>=90%) with the standard reference set of m^5C sites.

Performance Optimization: Once the accuracy requirements are met, performance improvement will become a key focus in the evaluation process. The proposal should include detailed optimization strategies.

Workflow Example:

(1) Build indexes for the reference genome and non-coding RNA (ncRNA):

```
hisat-3n/hisat-3n-build -p 32 --base-change C,T /asc25/ref/Homo_sapiens.GRCh38.dna.primary_assembly.fa /asc25/ref/Homo_sapiens.GRCh38.dna.primary_assembly.fa

samtools-1.21/samtools faidx /asc25/ref/Homo_sapiens.GRCh38.dna.primary_assembly.fa

awk 'BEGIN{{OFS="\\t"}}{{print $1,$1,0,$2,"+"}}' /asc25/ref/Homo_sapiens.GRCh38.dna.primary_assembly.fa.fai >Homo_sapiens.GRCh38.dna.primary_assembly.fa.saf

hisat-3n/hisat-3n-build -p 16 --base-change C,T /asc25/ncrna_ref/Homo_sapiens.GRCh38.ncrna.fa /asc25/ncrna_ref/Homo_sapiens.GRCh38.ncrna.fa

samtools-1.21/samtools faidx /asc25/ncrna_ref/Homo_sapiens.GRCh38.ncrna.fa
```

(2) The data processing workflow (stage 1) for one of the samples:

cutseq /asc25/SRR23538290/SRR23538290.fastq -t 20 -A INLINE -m 20 --trim-polyA --ensure-inline -barcode -o /asc25/SRR23538290/SRR23538290.fastq_cut -s /asc25/SRR23538290/SRR23538290.fastq_tooshort -u /asc25/SRR23538290/SRR23538290.fastq_untrimmed

hisat-3n/hisat-3n --index /asc25/ncrna_ref/Homo_sapiens.GRCh38.ncrna.fa --summary-file /asc25/SRR23538290/map2ncrna.output.summary --new-summary -q -U /asc25/SRR23538290/SRR23538290.fastq_cut -p 16 --base-change C,T --mp 8,2 --no-spliced-alignment --directional -mapping | /asc25/samtools-1.21/samtools view -@ 16 -e '!flag.unmap' -O BAM -U /asc25/SRR23538290/SRR23538290.ncrna.unmapped.bam -o /asc25/SRR23538290/SRR23538290.ncrna.mapped.bam

samtools-1.21/samtools fastq -@ 16 -O /asc25/SRR23538290/SRR23538290.ncrna.unmapped.bam >/asc25/SRR23538290/SRR23538290.mRNA.fastq

hisat-3n/hisat-3n --index /asc25/ref/Homo_sapiens.GRCh38.dna.primary_assembly.fa -p 16 --summary-file /asc25/SRR23538290/map2genome.output.summary --new-summary -q -U /asc25/SRR23538290/SRR23538290.mRNA.fastq --directional-mapping --base-change C,T --pen-noncansplice 20 --mp 4,1 | samtools-1.21/samtools view -@ 16 -e '!flag.unmap' -O BAM -U /asc25/SRR23538290/SRR23538290.mRNA.genome.unmapped.bam -o /asc25/SRR23538290/SRR23538290.mRNA.genome.mapped.bam

samtools-1.21/samtools sort -@ 16 --write-index -O BAM -o /asc25/SRR23538290/SRR23538290.mRNA.genome.mapped.sorted.bam /asc25/SRR23538290/SRR23538290.mRNA.genome.mapped.bam

samtools-1.21/samtools view -@ 20 -F 3980 -c /asc25/SRR23538290/SRR23538290.mRNA.genome.mapped.sorted.bam >/asc25/SRR23538290/SRR23538290.mRNA.genome.mapped.sorted.bam.tsv

java -server -Xms8G -Xmx40G -Xss100M -Djava.io.tmpdir=/asc25/SRR23538290 -jar /asc25/UMICollapse-1.0.0/umicollapse.jar bam -t 2 -T 16 --data naive --merge avgqual --two-pass -i /asc25/SRR23538290/SRR23538290.mRNA.genome.mapped.sorted.bam -o /asc25/SRR23538290/SRR23538290.mRNA.genome.mapped.sorted.dedup.bam > /asc25/SRR23538290/SRR23538290.mRNA.genome.mapped.sorted.dedup.log

samtools-1.21/samtools index -@ 8 /asc25/SRR23538290/SRR23538290.mRNA.genome.mapped.sorted.dedup.bam /asc25/SRR23538290/SRR23538290.mRNA.genome.mapped.sorted.dedup.bam.bai

samtools-1.21/samtools view -e "rlen<100000" -h /asc25/SRR23538290/SRR23538290.mRNA.genome.mapped.sorted.dedup.bam | hisat-3n/hisat-3n-table -p 16 -u --alignments - --ref /asc25/ref/Homo_sapiens.GRCh38.dna.primary_assembly.fa --output-name /dev/stdout --base-change C,T | cut -f 1,2,3,5,7 | gzip -c > /asc25/SRR23538290/SRR23538290_unfiltered_uniq.tsv.gz

```
samtools-1.21/samtools view -e "rlen<100000" -h /asc25/SRR23538290/SRR23538290.mRNA.genome.mapped.sorted.dedup.bam | hisat-3n/hisat-3n-table -p 16 -m --alignments - --ref /asc25/ref/Homo_sapiens.GRCh38.dna.primary_assembly.fa --output-name /dev/stdout --base-change C,T | cut -f 1,2,3,5,7 | gzip -c > /asc25/SRR23538290/SRR23538290_unfiltered_multi.tsv.gz

samtools-1.21/samtools view -@ 8 -e "[XM] * 20 <= (qlen-sclen) && [Zf] <= 3 && 3 * [Zf] <= [Zf] + [Yf]" /asc25/SRR23538290/SRR23538290.mRNA.genome.mapped.sorted.dedup.bam -O BAM -o /asc25/SRR23538290/SRR23538290.mRNA.genome.mapped.sorted.dedup.filtered.bam

samtools-1.21/samtools view -e "rlen<100000" -h /asc25/SRR23538290/SRR23538290.mRNA.genome.mapped.sorted.dedup.filtered.bam | /asc25/hisat-3n/hisat-3n-table -p 16 -u --alignments - --ref /mnt/nvme2n1/asc25/ref/Homo_sapiens.GRCh38.dna.primary_assembly.fa --output-name /dev/stdout --base-change C,T | cut -f 1,2,3,5,7 | gzip -c > /asc25/SRR23538290/SRR23538290_filtered_uniq.tsv.gz

samtools-1.21/samtools view -e "rlen<100000" -h /asc25/SRR23538290/SRR23538290.mRNA.genome.mapped.sorted.dedup.filtered.bam | hisat-3n/hisat-3n-table -p 16 -m --alignments - --ref /asc25/ref/Homo_sapiens.GRCh38.dna.primary_assembly.fa --output-name /dev/stdout --base-change C,T | cut -f 1,2,3,5,7 | gzip -c > /asc25/SRR23538290/SRR23538290_filtered_multi.tsv.gz

python m5C-UBSseq-main/bin/join_pileup.py -i /asc25/SRR23538290/SRR23538290_unfiltered_uniq.tsv.gz /asc25/SRR23538290/SRR23538290_unfiltered_multi.tsv.gz /asc25/SRR23538290/SRR23538290_filtered_uniq.tsv.gz /asc25/SRR23538290/SRR23538290_filtered_multi.tsv.gz -o /asc25/SRR23538290/SRR23538290_genome.arrow
```

The above workflow needs to be executed for **each** dataset.

(3) The data processing workflow (stage 2):

```
python /asc25/m5C-UBSseq-main/bin/group_pileup.py -i ./SRR23538290/SRR23538290_genome.arrow ./SRR23538291/SRR23538291_genome.arrow ./SRR23538292/SRR23538292_genome.arrow -o WT.arrow

python m5C-UBSseq-main/bin/select_sites.py -i ./WT.arrow -o ./WT.prefilter.tsv

python ./m5C-UBSseq-main/bin/filter_sites.py -i ./SRR23538290/SRR23538290_genome.arrow -m ./WT.prefilter.tsv -b ./SRR23538290/SRR23538290.bg.tsv -o ./SRR23538290/SRR23538290.filtered.tsv

python ./m5C-UBSseq-main/bin/filter_sites.py -i ./SRR23538291/SRR23538291_genome.arrow -m ./WT.prefilter.tsv -b ./SRR23538291/SRR23538291.bg.tsv -o ./SRR23538291/SRR23538291.filtered.tsv

python ./m5C-UBSseq-main/bin/filter_sites.py -i ./SRR23538292/SRR23538292_genome.arrow -m ./WT.prefilter.tsv -b ./SRR23538292/SRR23538292.bg.tsv -o ./SRR23538292/SRR23538292.filtered.tsv
```

Note

(1) Any changes to the original sequencing data files are strictly prohibited.

(2) The final output files and log files must remain unmodified.

(3) Optimize and minimize the elapsed time of the workflow as much as possible on the premise that the process and results are correct.

For any further questions, please contact techsupport@asc-events.org

--End--