Click to verify

STAR, or Spliced Transcripts Alignment to a Reference, is a bioinformatics tool designed to align high-throughput RNA-seq data. This technique studies the transcriptome, comprising all RNA molecules, including mRNA, rRNA, tRNA, and non-coding RNAs produced in cells. ========== STAR is a versatile and efficient tool for RNA-seq data analysis, allowing for fast and accurate alignment of RNA sequences. This two-pass mode allows STAR to perform a more sensitive alignment by using splice junctions discovered in the first pass for the second pass. ## Step 1: Setting up the STAR Index Before running the alignment, STAR must generate its own format of index based on the genomic information provided. To do this, create a new working directory and a subdirectory for your STAR index. mkdir ~/star && cd ~/star mkdir STARIndex ## Step 2: Generating the STAR Index Generate a STAR index based on the mm10 FASTA and your GTF files using the following command: STAR \ --runMode genomeGenerate \ --genomeDir STARIndex \ --genomeFastaFiles /projects/micb405/analysis/STAR_tutorial/Mus_musculus.GRCm38.dna.primary_assembly.fa \ --sjdbGTFfile /projects/micb405/analysis/STAR_tutorial/Mus_musculus.GRCm38.84.gtf \ --sjdbOverhang 49 \ --runThreadN 16 ## Step 3: Downloading RNA-seq FASTQ Files Download healthy tissue RNAseq FASTQ files from the paper “An RNA-Seq atlas of gene expression in mouse and rat normal tissues”. These files are located in the associated ArrayExpress that can be found in the data citations from the article’s NCBI page. ## Step 4: Running STAR for Each Sample When the STAR index is ready, run STAR, outputting into a separate directory for each sample you wish to align. You can control the output directory with the string provided to --outFileNamePrefix. Create a script that runs this STAR command for each of your samples and run it in a way so that it won’t stop when you exit your shell. STAR \ --genomeDir STARIndex/ \ --readFilesIn sample1.fastq.gz \ --outFileNamePrefix sample1/sample1.fastq.gz \ --runThreadN 8 \ --limitBAMsortRAM 60000000000 \ --outSAMattrRGline ID:sample1.fastq.gz SM:sample1.fastq.gz \ --outBAMsortingThreadN 8 \ --outSAMtype BAM SortedByCoordinate \ --outSAMunmapped Within \ --outSAMstrandField intronMotif \ --readFilesCommand zcat \ --chimSegmentMin 20 \ --genomeLoad NoSharedMemory ========== How Do You Choose Samples and Run a Script for STAR Alignment? The STAR (Spliced Transcripts Alignment to a Reference) aligner is used to map sequence reads to a reference genome. In this chapter, we'll explore how to choose samples and run the STAR script. Choosing Samples We need to select our sample files to test with the STAR script. To do this, create a script that performs the STAR alignment for your files. Running the STAR Script The book "MICB 405 Bioinformatics: 2021.22" by Axel Hauduc and Stephan Koenig provides instructions on how to run the STAR script. The book was last built on February 17, 2022, using the bookdown R package. The approximate time required for this exercise is around 90 minutes. Learning Objectives To get started with STAR alignment, we need to understand: 1. The alignment method used by STAR. 2. The intricacies of alignment tools used in NGS analysis (parameters and usage). 3. Choosing appropriate STAR alignment parameters for our dataset. After exploring the quality of our raw reads, we can move on to read alignment using the STAR aligner. Read Alignment Read alignment determines where in the genome our reads originated from. We need to choose an appropriate reference genome to map our reads against and perform the read alignment using one of several splice-aware alignment tools such as STAR or HISAT2. The choice of aligner is often a personal preference, and it's also dependent on the computational resources available to you. STAR Aligner We'll use the STAR aligner to determine where on the human genome our reads originated from. The STAR algorithm achieves high accuracy and outperforms other aligners by more than a factor of 50 in mapping speed but is memory-intensive. The algorithm uses a two-step process: 1. Seed searching: For every read that STAR aligns, it searches for the longest sequence that exactly matches one or more locations on the reference genome. 2. Clustering, stitching, and scoring: The separate seeds are stitched together to create a complete read by first clustering the seeds together based on proximity to a set of 'anchor' seeds. Running the STAR Script To get started with this lesson, start an interactive session with 6 cores: `srun --pty -p short -t 0-12:00 -c 6 --mem 8G --reservation=HBC /bin/bash` You should have a directory tree setup similar to that shown below. It's best practice to have all files you intend on using for your workflow present within the same directory. The book provides instructions on how to load the module and use the STAR aligner: `$ module load gcc/6.2.0 star/2.5.2b` Aligning reads using STAR involves loading the required modules, setting up the working directory, and running the script with 6 cores and 8GB of memory. Conclusion By following these instructions, you should be able to choose samples and run a script for STAR alignment. The book provides detailed instructions on how to use the STAR aligner, including setup and usage guidelines.Creating a genome index is an essential step in mapping reads to a reference genome. This process involves creating a directory to store the indices and generating them using the STAR tool. The basic options for generating genome indices include specifying the number of threads, mode, genome directory, FASTA file, GTF file, and read length. To create a job submission script, add the necessary SLURM directives and the STAR command. The script should specify the partition name, run time limit, number of cores, memory, and job name. The STAR command should include the input files, genome directory, and output prefix. After generating the genome indices, aligning reads can be performed using the STAR tool. The basic options for aligning reads include specifying the number of threads, input FASTQ file, genome directory, and output prefix. Additional parameters such as output filetype, unmapped reads, and maximum multiple alignments can also be specified. For instance, to align a single sample, one can use the following command: ``` STAR --runThreadN 6 \ --readFilesIn /path/to/FASTQ_file \ --genomeDir /path/to/genome_indices_directory \ --outFileNamePrefix prefix ``` By understanding and applying these options and parameters, researchers can efficiently map reads to a reference genome using STAR.STAR Aligner Utilizes Efficient Two-Step Process for RNA-seq Mapping ========== We can access the STAR command by simply using the STAR command followed by the basic parameters described above and any additional parameters. The full command is provided below for you to copy paste into your terminal. If you want to manually enter the command, it is advisable to first type out the full command in a text editor (i.e. Sublime Text or Notepad++) on your local machine and then copy paste into the terminal. This will make it easier to catch typos and make appropriate changes. STAR --genomeDir /n/groups/hbctraining/intro_rnaseq_hpc/reference_data_ensembl38/ensembl38_STAR_index/ \ --runThreadN 6 \ --readFilesIn Mov10_oe_1.subset.fq \ --outFileNamePrefix ../results/STAR/Mov10_oe_1_ \ --outSAMtype BAM SortedByCoordinate \ --outSAMunmapped Within \ --outSAMattributes Standard This lesson has been developed by members of the teaching team at the Harvard Chan Bioinformatics Core (HBC). These are open access materials distributed under the terms of the Creative Commons Attribution license (CC BY 4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited. Approximate time: 90 minutes Learning Objectives: Understanding the alignment method STAR utilizes to align sequence reads to the reference genome Identifying the intricacies of alignment tools used in NGS analysis (parameters, usage, etc) Choosing appropriate STAR alignment parameters for our dataset Read Alignment Now that we have explored the quality of our raw reads, we can move on to read alignment. We perform read alignment or mapping to determine where in the genome the reads originated from. The alignment process consists of choosing an appropriate reference genome to map our reads against and performing the read alignment using one of several splice-aware alignment tools such as STAR or HISAT2. The choice of aligner is often a personal preference and also dependent on the computational resources that are available to you. STAR Aligner To determine where on the human genome our reads originated from, we will align our reads to the reference genome using STAR (Spliced Transcripts Alignment to a Reference). STAR is an aligner designed to specifically address many of the challenges of RNA-seq data mapping using a strategy to account for spliced alignments.star set-up tutorial ========== to begin working on this lesson start an interactive bash session with six cores you should have a directory tree similar to the one below its best practice to have all files used in your workflow present within the same directory we have our original fastq files generated earlier rnaseq ├── logs ├── meta ├── raw_data │ ├── Irrel_kd_1.subset.fq ├── Irrel_kd_2.subset.fq ├── Irrel_kd_3.subset.fq ├── Mov10_oe_1.subset.fq ├── Mov10_oe_2.subset.fq │ └── Mov10_oe_3.subset.fq ├── results └── scripts to use the star aligner load the module srun --pty -p short -t 0-12:00 -c 6 --mem 8G --reservation=HBC /bin/bash ========== aligning reads using star is a two-step process creating a genome index map reads to the genome a quick note on shared databases for human and other commonly used model organisms the o2 cluster has a designated directory at in which there are files that can be accessed by any user these files contain but are not limited to genome indices for various tools reference sequences tool specific data and data from public databases such as ncbi and pdb ========== so when using a tool that requires a reference of sorts it is worth taking a quick look here because chances are its already been taken care of for you ls -l /n/groups/shared_databases/igenome creating a genome index for this workshop we are using reads that originate from a small subsection of chromosome 1 ~300000 reads and so we are using only chr1 as the reference genome to store our genome indices we will use the nscratch2 space with large temporary storage capacity we need to create a directory for the indices within this space mkdir -p /n/scratch2/username/chr1_hg38_index ========== the basic options to generate genome indices using star are as follows --runThreadN number of threads --runMode genomeGenerate mode --genomeDir path/to/store/genome_indices --genomeFastaFiles path/to/FASTA_file --sjdbGTFfile path/to/GTF_file --sjdbOverhang readlength -1 note in case of reads of varying length the ideal value for --sjdbOverhang is max(readLength)-1 ========== now lets create a job submission script to generate the genome index vim ~/unix_lesson/rnaseq/scripts/genome_index.run within vim we now add our shebang line the slurm directives and our star command #!/bin/bash #SBATCH -p short # partition name #SBATCH -t 0-2:00 # hoursminutes runlimit after which job will be killed #SBATCH -c 6 # number of cores requested -- this needs to be greater than or equal to the number of cores you plan to use to run your job #SBATCH --mem 16G #SBATCH --job-name STAR_index # Job name #SBATCH -o %j.out # File to which standard out will be written #SBATCH -e %j.err # File to which standard err will be written cd /n/scratch2/username/ module load gcc/6.2.0 star/2.5.2b star --runThreadN 6 \ --runMode genomeGenerate \ --genomeDir chr1_hg38_index \ --genomeFastaFiles /n/groups/hbctraining/intro_rnaseq_hpc/reference_data_ensembl38/Homo_sapiens.GRCh38.dna.chromosome.1.fa \ --sjdbGTFfile /n/groups/hbctraining/intro_rnaseq_hpc/reference_data_ensembl38/Homo_sapiens.GRCh38.92.gtf \ --sjdbOverhang 99 $ sbatch ~/unix_lesson/rnaseq/scripts/genome_index.run ========== aligning reads after you have the genome indices generated you can perform the read alignment we previously generated the genome indices for you in /n/groups/hbctraining/intro_rnaseq_hpc/reference_data_ensembl38/ensembl38_STAR_index/ directory so that we dont get held up waiting on the generation of the indices create an output directory for our alignment files cd ~/unix_lesson/rnaseq/raw_data $ mkdir ../results/STARSTAR --genomeDir /n/groups/hbctraining/intro_rnaseq_hpc/reference_data_ensembl38/ensembl38_STAR_index/ \ --runThreadN 6 \ --readFilesIn Mov10_oe_1.subset.fq \ --outFileNamePrefix ../results/STAR/Mov10_oe_1_ \ --outSAMtype BAM SortedByCoordinate \ --outSAMunmapped Within \ --outSAMattributes Standard ========== Looking forwrd to runnin STAR on our data! We'll be usin the reference genome /n/groups/hbctraining/intro_rnaseq_hpc/reference_data_ensembl38/ensembl38_STAR_index/ and settin our thread numbr to 6. The output wil be savd in ../results/STAR/Mov10_oe_1_. We'll also sort the alignments by coordniate and exclude unmapped reads. ========== Now dat we hav explord our raw read quiality, we can move on to read alignment. We'll align our reads to the refernce genome usin STAR (Spliced Transcripts Alignment to a Referance). STAR is a good choise for RNA-seq data mapping, but it's memory intensive. ========== STAR uses a twostep process to achive its high accuresy. First, it searchs for the longest sequnce that exaktly matches one or more locashuns on the refernce genome. This is calld the Maximal Mappable Prefixes (MMP). Then, it searchs agen for only the unmapped porshun of the read to find the next longest sequnce. ========== STAR is fast and accures, but it's also memory intensive. It ush the uncompressed suffix array (SA) to eficiently search for the MMPs, whish allows for quick searchin against even the largist refernce genomes.To initiate STAR sequencing analysis, a two-step process is required. The initial step involves creating a genome index and mapping reads to the genome. ========== Running the STAR aligner requires loading the necessary modules before proceeding with the alignment. This can be achieved by utilizing the SLURM directives for job submission and resource allocation. A recommended approach includes creating a dedicated directory tree for storing files related to the workflow, including original FASTQ files. ========== Creating a genome index is crucial for STAR sequencing analysis. The process involves specifying several key options, including the number of threads (--runThreadN), genome generation mode (--runMode), and the path to store genome indices (--genomeDir). Furthermore, the --genomeFastaFiles option should be set to point towards the FASTA file used as a reference sequence. ========== To ensure successful genome indexing, it is essential to create a job submission script that incorporates necessary directives such as partition name, run time limit, and core allocation. This script will facilitate the execution of the STAR aligner command within SLURM, which in turn enables efficient resource utilization and optimized workflow management. ========== After creating the genome index, the next step involves aligning reads to the genome. This process relies heavily on the previously generated indices, allowing for accurate mapping and sequencing analysis. By leveraging pre-existing databases and shared directories, researchers can streamline their workflow and minimize potential issues associated with missing or inadequate reference sequences.The importance of creating an output directory cannot be overstated in order to avoid any delays in the generation of indices. We need to create a designated area for our alignment files: $ cd ~/unix_lesson/rnaseq/raw_data $ mkdir ../results/STAR For the purpose of this exercise, we will focus on a single sample and set up our workflow accordingly. To begin with, we will utilize the first replicate in the Mov10 over-expression group, which is represented by the file Mov10_oe_1.subset.fq. The user manual for STAR provides detailed information on its functionality and various options available. The fundamental options for aligning reads to a genome using STAR are as follows: --runThreadN: specifying the number of threads or cores --readFilesIn: providing the path to the FASTQ file --genomeDir: indicating the location of the genome indices directory --outFileNamePrefix: assigning a prefix to all output files In addition, we will employ several supplementary parameters in our command. These include --outSAMtype: setting the output filetype to SAM (default), --outSAMunmapped: determining what to do with unmapped reads Note that STAR's default settings are optimized for mammalian genomes; therefore, it is essential to consider species-specific requirements and modify alignment parameters as necessary. The provided command can be copied and pasted into the terminal, but we strongly advise using a text editor first to avoid any typos or errors. The full command is: STAR --genomeDir /n/groups/hbctraining/intro_rnaseq_hpc/reference_data_ensembl38/ensembl38_STAR_index/ \ --runThreadN 6 \ --readFilesIn Mov10_oe_1.subset.fq \ --outFileNamePrefix ../results/STAR/Mov10_oe_1_ \ --outSAMtype BAM SortedByCoordinate \ --outSAMunmapped Within \ --outSAMattributes Standard This lesson is the result of collaboration between members of the teaching team at the Harvard Chan Bioinformatics Core (HBC). The materials are available under a Creative Commons Attribution license (CC BY 4.0), allowing unrestricted use, distribution, and reproduction in any medium. We will now proceed to read alignment, which involves mapping sequence reads against a reference genome using an appropriate aligner such as STAR or HISAT2.The Maximal Mappable Prefixes (MMPs) are the longest matching sequences between a read and the reference genome. These MMPs serve as 'seeds' for further mapping of the read. The STAR algorithm searches for these seeds sequentially, starting with seed1, which is mapped to the genome, and then searching for the next longest sequence, seed2. STAR utilizes an uncompressed suffix array (SA) for efficient searching of MMPs. This approach allows for rapid alignment against even the largest reference genomes. In contrast, other aligners often search for the entire read sequence before splitting reads and performing iterative rounds of mapping. If STAR fails to find an exact match for each part of the read due to mismatches or insertions/deletions (indels), the previous MMPs are extended. If this extension does not yield a good alignment, the poor-quality or adapter sequence is soft-clipped. ========== mkdir -p /share/ScratchGeneral/[your_ID]/STAR_output/ cd /share/ScratchGeneral/[your_ID]/STAR_output/ STAR command in interactive bash The user manual for STAR provides detailed information on its functionality and available options. To align reads to the genome, you can use the following basic parameters: --runThreadN: number of threads / cores --readFilesIn: /path/to/FASTQ_file --genomeDir: /path/to/genome_indices_directory --outFileNamePrefix: prefix for all output files We will be using additional parameters such as --outSAMtype, --outSAMunmapped. Default filtering is applied to limit the maximum number of multiple alignments allowed for a read to 10. We can access the software by using the STAR command with the basic parameters described above and any additional parameters. The full command is provided below for you to copy paste into your terminal. RUN_MODE="alignReads" SAMPLE="/share/ScratchGeneral/[your_ID]/rnaseq_tutorial/TRIMMED_FASTA/SRR306844chr1_chr3.trimmed.fastq.gz" OUT_PREFIX="SRR306844chr1_chr3" SAM_TYPE="BAM SortedByCoordinate" GENOME_DIR="/share/ScratchGeneral/[your_ID]/rnaseq_tutorial/STAR/" STAR --runMode "${RUN_MODE}" \ --genomeDir ${GENOME_DIR} \ --runThreadN 6 \ --readFilesIn ${SAMPLE} \ --outFileNamePrefix ${OUT_PREFIX} \ --outSAMtype ${SAM_TYPE} To automate this command across every sample, you can use a for loop similar to the trimmomatic script.Data Upload in Galaxy Tutorial ========== Looking forward to uploading our data in galaxy tomorrow and discussin our strategies. Using Galaxy, RStudio Cloud, and Miniconda to Analyze Sequencing Data in a Pipeline ========== Registering for RStudio Cloud or logging in creates a new project. In our scenario, we're utilizing an existing environment within RStudio Cloud because miniconda is pre-installed, allowing us to install essential packages using conda create -n env_name package_list. We are going to use the terminal provided by RStudio to process thousands of data files. Our workspace provides a left-terminal window that has just opened. We will utilize it for subsequent code execution. To get started, we activate our newly created environment and proceed with installing necessary packages such as fastqc, cutadapt, star, samtools, and subread. ## Step 1: Installing Required Packages We need to install the required packages in our environment using conda create -n env_name package_list. This includes fastqc, cutadapt, star, samtools, and subread. ## Step 2: Activating the Environment Once installed, we activate our newly created environment so that its packages can be used in subsequent commands. We use $ conda activate name_of_your_env for this purpose. Here, "name_of_your_env" is the actual name of your environment as specified during the creation process. ## Step 3: Running Software Version Manual We need to make sure we are using the correct version of these packages. For that, we will open their respective manual pages and get familiar with their usage. fastqc has a manual page for its usage in Linux, MacOS, and Windows environments. Similarly, cutadapt also has its own manual page that provides detailed information about how to use it. ## Step 4: Running Manual Available for Description We can check the version of these packages using the command $ package_name --version. For example, we would enter "fastqc" followed by "--version". This will output our current version number. Similarly, other commands such as STAR and SAMtools also have their respective manual pages that describe how to use them. ## Step 5: Installing Subread and Its Packages We now go ahead with installing subread, which is a general-purpose RNA-seq aligner. It comes with featureCounts, a highly efficient read summarization program. For this purpose, we will again utilize the conda create -n env_name package_list command to install these packages. ## Step 6: Getting Started in RStudio Now that we have our environment set up and necessary packages installed, we proceed by using them within RStudio Cloud for further data analysis. ## Step 7: Data Importing Through gx_get Function We import files from Galaxy through the "gx_get" function. This allows us to access the data imported into our RStudio project. Here's how it works: ## Step 8: Quality Control Measures During sequencing, errors occur due to various technical limitations of each platform. These errors might bias analysis and affect interpretation of data. ## Step 9: FastQC Tool Usage To address this issue, we use fastqc tools to generate reports on sequence quality. fastqc also includes an option that allows us to save our results in a specified folder for future assessment and reference. We can access these saved files from within RStudio and assess their quality. ## Step 10: Trim Reads Using Cutadapt We then trim the reads using cutadapt to get rid of bases sequenced with low uncertainty at read ends and remove overall bad-quality reads. This step allows us to restore trimmed file names, making further processing easier in the future. We save these results in a separate folder called "trimmedData" within our RStudio project.FastQC and STAR: A Step-by-Step Guide for Genomic Data Analysis ========== Looking forward to performing quality checks on our trimmed data files located in the trimmedData folder tomorrow and discussing our strategies. We perform read alignment or mapping to determine where in the genome our reads originated from. There are a number of tools to choose from and, while there is no gold standard, some tools are better suited for particular NGS analyses. We will be using the STAR Aligner (STAR), which is a universal aligner for mapping sequences against a large reference genome. Our first step is to index the reference genome for use by STAR. Indexing allows the aligner to quickly find potential alignment sites for query sequences in a genome, which saves time during alignment. This process takes some time and will create an index that can be reused with different tools or genomes. After indexing the reference genome, we'll start by aligning the reads from just one of our samples (GSM461177) to it using STAR. We'll then iterate this process on the second pair of our sample files (GSM461180). The alignment process consists of choosing an appropriate reference genome and then deciding on an aligner. The aligned reads will be stored in a SAM file, which is a tab-delimited text file containing information for each individual read and its alignment to the genome. We can compress this binary version into a BAM file for efficient random access and indexing. We'll need to make sure we're using the correct parameters for our STAR command, as it may vary depending on the specific use case.The alignment information corresponds to the essential mapping details for a single read. Each alignment line has 11 mandatory fields for vital mapping information and an additional variable number of fields for aligner-specific data. An example entry from a SAM file is shown below, with different fields highlighted. ========== We will convert the SAM file to BAM format using the samtools program with the view command, instructing this command that the input is in SAM format (-S) and to output BAM format (-b). We begin by processing the first sample (GSM461177), then repeat the process for the second sample (GSM461180): $ samtools view -S -b GSM461177Aligned.out.sam > GSM461177Aligned.out.bam [samopen] SAM header is present: 1 sequences. Next, we sort the BAM file using the sort command from samtools, specifying output directory (-o). $ samtools sort -o GSM461177Aligned.out.sorted.bam GSM461177Aligned.out.bam Our files are small, so we will not see this output if you run the workflow with larger files. SAM/BAM files can be sorted in multiple ways, e.g. location on the chromosome, read name, etc. It is essential to recognize that different alignment tools yield differently sorted SAM/BAM files, and various downstream tools require differently sorted alignment files as input. You may use samtools to learn more about this BAM file as well. Time has come to conclude! To compare gene expression between conditions (e.g. with or without PS depletion), we must quantify the number of reads per gene or more precisely, the number of reads mapping to exons of each gene. For the final step of this tutorial, we utilize featureCounts to count reads per annotated gene. The main output is a table containing counts - i.e., the number of reads (or fragments in paired-end reads) mapped to each gene (in rows, with their ID in the first column) in the provided annotation. FeatureCount also generates feature length datasets. ========== ========== To run the STAR analysis, you will need a computer with sufficient memory and disk space. Alternatively, you can use high-performance computers (HPCs) if available. If not, ensure your machine meets the following specifications: Linux or Mac OS Sufficient RAM (at least 32 GB for human-sized genomes) A hard drive with sufficient disk space (recommended 100 to 500 GB based on input file sizes) Multiple processors (cores) for parallel computation Install STAR by downloading pre-compiled binaries or building it from source. For Linux, use: git clone export PATH=$PATH:/home/ren/software/STAR/bin/Linux_x86_64 verify the added binaries. The STAR software includes two main steps for mapping reads to a reference genome: 1. Building a reference genome index using FASTA and GTF/GFF3 files. 2. Mapping RNA-seq reads (FASTQ or FASTA) to the indexed genome. For building the genome index, STAR uses the provided genome file and gene annotation file (GTF or GTF/GFF3). The gene annotation file is necessary for creating known splice junctions to improve accuracy. STAR version = 2.7.10b build genome index using GTF file: STAR --runThreadN 12 --runMode genomeGenerate --genomeDir ath_star_index --genomeFastaFiles Athaliana_TAIR10.fasta --sjdbGTFfile Athaliana_gene.gtf --sjdbOverhang 149 Other parameters for building genome indices include: --runThreadN: Number of threads (processors) for mapping reads to a genome --runMode: Run mode, where genomeGenerate builds genome index --genomeDir: Path to the directory where genome indices are stored --genomeFastaFiles: Reference genome file in FASTA format --sjdbGTFfile: GTF file for gene annotation (optional) --sjdbOverhang: Length of reads around splice junctions If you don't have a GTF annotation file, consider using a GFF3 file instead. Ensure to specify the parent-child relationship with --sjdbGTFtagExonParentTranscript Parent. Once genome indices are created, map single-end or paired-end RNA-seq reads to the reference genome using STAR: For single-end reads: STAR version=2.7.10b # map single end reads to genome --runThreadN 12 --readFilesIn ath_seed_sample.fastq --genomeDir ath_star_index --outSAMtype BAM SortedByCoordinate --outFileNamePrefix seed_sample --outSAMunmapped Within For paired-end reads: STAR version=2.7.10b # map paired-end reads to genome --runThreadN 12 --readFilesIn ath_sample_read1.fastq ath_sample_read2.fastq --genomeDir ath_star_index --outSAMtype BAM SortedByCoordinate --outFileNamePrefix seed_sample --outSAMunmapped Within If your read files are gzip compressed, use the --readFilesCommand parameter to decompress them. ========== The STAR (Spliced Transcripts Alignment to a Reference) software is a widely used tool for mapping high-throughput sequencing data to genomes. To utilize STAR, users need to specify several parameters that describe the reads and the genome being targeted. STAR Parameters Description **Mapping Reads** * **Number of threads**: This parameter specifies the number of threads (processors) that will be utilized by the STAR software to perform the mapping. A higher number of threads can significantly speed up the processing time. * **Read files**: Users must provide read files for mapping to the genome. For paired-end reads, both `read1` and `read2` files are required. If there are multiple samples, separate files should be provided by a comma. Genome Information * **Genome directory**: This parameter points to the location of the built genome indices that will be used for mapping. * **Output coordinate sorted BAM file**: Users can optionally specify an output BAM file that is sorted by coordinate. This is useful for many downstream analyses. **Advanced Parameters** * **Unmapped reads from main SAM file**: This parameter allows users to save unmapped reads in the main SAM file in a separate format. * **Output file prefix**: A prefix can be specified for the output files. Novel Splice Junctions Analysis If the goal is to identify novel splice junctions, such as those involved in differential splicing analysis, it is recommended to use 2-pass mapping with re-building of genome indices using splice junction information. STAR Output Files After successful mapping, STAR generates several output files that contain the mapped reads. These include: * Alignment in BAM format (sorted by coordinate) * Alignment summary statistics * Alignment log for commands and parameters (useful for troubleshooting) * Alignment progress report * Filtered splice junctions found during the mapping stageUsing STAR for 2-Pass Mapping in Bioinformatics To enhance your skills in bioinformatics and genomics, consider taking courses that focus on these fields. ========== Several RNA-seq alignment tools have been evaluated based on experimental data from the model plant Arabidopsis thaliana, including STAR. STAR: ultrafast universal RNA-seq aligner has been widely used for its ability to align RNA-seq reads quickly and accurately. A simulation-based comprehensive benchmarking of RNA-seq aligners was conducted to compare the performance of different tools. For any questions, comments or recommendations regarding this article, please email me at reneshbe@gmail.com ==========