Lukas UMI-tools & MAUDE pipeline

Steps 2, 3, and 4 of this pipeline are for single-end reads **only**!
Steps 1 and 5 can work for paired end reads as well.

If you got UMI counts from cellranger, bcwithqc, or had John generate counts for you, you can skip step 2, 3, and 4.

Currently only supports three bin sorting (Upper & Lower + Input/Unsorted)!

0. File Name Conventions

- Your fastq files must be compressed, and have the file ending ".fastq.gz".
  They must be named in the pattern "X_L$_#" (example: "L_L1_4.fastq.gz").
- If you got UMI counts from cellranger, bcwithqc, or had John generate counts for you, the "*_output_all" files must start with the pattern "XL$#" (example: "LL14_output_all")
- "X" can be "I", "L", "U" to signify the sorting bin: Input, Lower, and Upper
- "$" can be any number and specifies the sublibrary this sample is from.
- "#" can be any number and specifies the sample number.

1. Build the reference genome

1. Open the file "Combined_pipeline_support.Rmd".
2. Make sure all the packages are installed. If not use BiocManager::install("Package_Name")
3. In the setup block: change "first_time" to *TRUE* and "output_folder" to wherever you want your output folder to be. Then run the setup code_block.
4. Go to the read_raw block, change the raw_path to the path where your fastq file is located, and then then run the "read_raw" block to have a look at your reads. This will also tell you the most common read length, which will be used for calculating "SJDB_OVERHANG" later.
5. **If you are using the HD whole genome sgRNA library**:
   in the "load_tables" block, simply specify which sub-libraries (if any) you wish to exclude in "remove_these_sublibraries", and then run the block.
   If you are using a different sgRNA library:
   You need to change the entire "load_tables" block as specified below (or change the build_ref_df function from the make_reference_fasta code block to fit a differently formated dataframe).
   Load your sgRNA library, and format it so that at the end of the "load_tables" block you have a dataframe called "merged_sgRNA_df", which has:
   a. A "seq" column containing the sgRNA nucleotide sequence
   b. A "sgrna_id" column containing an unique identifier/name for the sgRNA. Make sure the name of nontargeting control sgRNAs starts with "CONTROL_C_NONTARG_" and that of targeting controls with "CONTROL_C_".
   c. A "entrez" column containing the entrez ID for the gene (or NA for controls).

d. A “symbol” column and a “Gene” column, both containing the gene symbol (for example GAPDH). For control sgRNAs those rows in the columns need to be NA (return *TRUE* for is.na())
e. A “sublib” column, specifying the sublibrary. These can later be used for replicates so set them to “ICS_#”, where # is your replicate number. If you do not have sublibraries or replicates, just set all entries to “ICS_1”.
f. A “src_lib” column specifying where you got the sgRNA from (doesn’t need to contain data, the column just needs to be there)
g. A “chr” column specifying which chromosome is targeted (doesn’t need to contain data, the column just need to be there).
h. A “count” column containing integers (doesn’t need to contain data, the column just need to be there).
i. A “start” and “end” column containing the start and end position of the target sequence on the chromosome (the columns don’t need to contain data, they just need to be there)
j. Remember to add this line saveRDS(merged_sgRNA_df, get_file_path(rds_output_folder, "merged_sgRNA_df.rds"))

6. In the “make_reference_fasta” code block:
change the past0() statement in the first line to fit your construct. If you are using the same modified CROP-Seq construct as Nico and Jia, you don’t need to change anything.
Currently it is set to sgRNA-sequence+Constant-Sequence_1. Do not include UMIs or staggers here!
Then run the code block. This will also calculate the value you should use for “genomeSAindexNbases” later on.

2. Run the cluster pipeline without read-filtering

1. Open the file “Combined_pipeline.sh”
2. Change all the cluster settings:
    a. Rename the job after “#SBATCH -J”
    b. If you are not from the Steinmetz group, change the group after “#SBATCH -A”
    c. Change the output paths of log files after “#SBATCH -o” and “#SBATCH -e”
    d. Change the email settings so it doesn’t send an email to me, after “#SBATCH --mail-type=” and “#SBATCH --mail-user=”
    e. (optional) Change the resources the job requests on the cluster after “#SBATCH --mem”, “#SBATCH -N”, “#SBATCH -n”, and “#SBATCH -t”
    f. (optional) I have included two copy paste lines in line 2 and 3 for chmod and sbatch, change the file path of those to your version of “Comined_pipeline.sh” or whatever you have renamed the file to. You can copy paste the chmod line into the cluster, to make the fiel executable, and the sbatch line to queue the file on the cluster after you have finished changing the rest of the options (there are still more of those after this step).
3. Change the User options:
    a. Change “INPUT_FOLDER” to wherever your compressed fastq files are located. They need to have the file ending “.fastq.gz”.

b. Change “OUTPUT_FOLDER” to your output folder. This needs to be the same folder you chose in step 1.2.
c. Change “REGEX_PATTERN” to the correct pattern for your reads. This requires understanding of regex & UMI-tools or trust in ChatGPT.
d. Change STAR settings:
    i. Change “SJDB_OVERHANG” and “Genome_SA_index_N_Bases” to the values you received earlier during step 1.3 and 1.5
    ii. If you changed “#SBATCH -n”, or “#SBATCH --mem” during step 2.2.e you now need to change NUM_THREADS and MAX_MEM respectively, to make sure they are not higher than the options you set.
e. Set Skip Options.
    If you plan to do read filtering later:
    Set “SKIP_FILTERING”, “SKIP_DEDUPLICATION”, and “SKIP_IDXSTATS” to *true* and all other skip options to *false*.
    If you do not intend to do read filtering at all:
    Set “SKIP_FILTERING” to *true* and all other skip options to *false*. After step 2.4 you can skip to step 5.1

4. Now you can execute the script on the cluster with “sbatch your_shell_script.sh”. Best to do this step overnight, as it may take a while, depending on the resources you provided, and the size of your files.

3. Setting Read-filtering Thresholds

1. Open the file “Combined_pipeline_support.Rmd”.
2. Scroll down to the “plot_reads_per_UMI” code block and run it. **This will take a while depending on file size**! It saves the graphs, so they do not need to be re-generated every time. Simply set “first_time” from the setup code block to *FALSE* (remember to re-execute the setup block), and it will load the finished graphs instead.
    To avoid clogging Rstudio you can also run the provided script “plot_reads_per_UMI.R” on the cluster, which will generate the same file, which will then be loaded when “first_time” is set to *FALSE*. All you need to do is change the “output_folder” in the script to your output folder (the same one as before).
3. Go to the “UMI_Knee_plot_2” code block and run it. This will print graphs for all your fastq files. Now you can inspect them to determine your thresholds.
4. Now you can manually set the threshold of how many reads per UMI you want to have as cutoff line. Go to the “generate_thresholds.tsv” codeblock, and manually change the “threshold_df” to your thresholds:
    a. In the “replicate” column you need to put the fastq file names without extension (If the file is “A_L2_4.fastq.gz” it would be “A_L2_4”)
    b. In the “threshold” column you need to put your custom cutoff points. The order needs to match the replicate column.
    c. Run the “generate_thresholds.tsv” codeblock
5. Run the “Filter_UMIs_with_few_reads” codeblock. This may take a while. There is a separate R script “filter_umis_with_few_reads” that can be run on the cluster to not clog Rstudio. Simply copy paste your threshold_df to the script and change the output_folder.

4. Run the pipeline with read-filtering

1. Open the file “Combined_pipeline.sh”
2. Under Skip Options, set “SKIP_FILTERING”, “SKIP_DEDUPLICATION”, and “SKIP_IDXSTATS” to *false*. Set all other “SKIP_” options to *true*
3. (optional) you can reduce the cluster resources for this part as in step 2.2.e
4. Execute the pipeline with sbatch on the cluster.

5. Data Analysis with MAUDE

1. Open the file “data_analysis_with_MAUDE.Rmd”
2. Make sure MAUDE is installed. See https://github.com/de-Boer-Lab/MAUDE
3. Make sure all the packages are installed. If not use BiocManager::install("Package_Name")
4. In the “setup” codeblock change the User Options
    a. change “first_time” to *TRUE*
    b. “output_folder” to the same you set during 1.2 and 2.3.b.
    c. **If you got your counts from cellRanger, bcwithqc, or from john**
        i. Set “pipeline” to “john”
        ii. Set “john_folder” to the folder containing “output_summary.txt” and all the subfolders for the individual fastqs.
    d. If there are files you want to skip, put their names in “skip_list”, otherwise set it to “skip_list <- c()”
    e. Set “method”:
        i. **“sum”** Treats all samples of the same sub-library as one and adds up their counts. Then treats all sublibraries as a single replicate. Best solution when individual samples have low coverage. Biased towards samples with higher counts.
        ii. **“rep”** Treats each sample & sublibrary combination as a replicate. Best solution when all samples have high coverage.
        iii. **“rep_sublib”** Combines all samples from a sublibrary into one replicate.
        iv. **“rep_sample”** Combines samples of the same number across sublibraries into one replicate. This might cause problems if the guides have different names in different sublibraries.
        v. **“”** Treats everything as one replicate. Simplest default solution.
    f. Set “norm_method”. This can be either “” for no normalization, or “control_median” for normalizing each sample based on the median of the non-targeting controls.
    g. Set “data_type” to “umis” unless you want to look at just the reads, without taking UMIs into consideration, in that case set it to “reads”.
    h. MAUDE requires equal amounts of data in each bin, if there is not enough data in the input bin, the pipeline generates data based on the averages of existing data. If you do not want this set “recover_input” to *FALSE*.
    i. Set “extra_suffix” if you plan to do analysis with multiple different settings to not overwritte names. Set it to “rf” if you want to apply simplified read filtering, for example if you are looking at just the reads. The threshold can be set with “simplified_rf_threshold”.
    j. Set “upper_lower_percentage” which percentage were the upper and lower bin gated to? (Example: Set to “0.10” for 10%)

    k. Set “same_controls_in_all_sublibraries”. Leave it as True for the HD whole genome library. Change it to FALSE only if each sublibrary/replicate has different control sgRNAs.
5. Now run the entire script. Since this may take a while, consider running the script as a job on the server.
6. Once it has finished, you can set “first_time” in the “setup” codeblock to FALSE, and it will now load the results from MAUDE instead of re-generating them.
7. Results:
    a. “maude_guide_stats” is a dataframe providing guide level stats
    b. “maude_gene_stats” is a dataframe providing gene level stats
    c. A CSV file in the output folder, containing all the Hits
    d. A CSV file in the output folder containing the top 100 Hits (Top 50 from either end of the curve)

6. Additional Data Exploration

The file “data_analysis_with_MAUDE.Rmd”, provides some additional code blocks for data exploration, which are not executed during Run All.

Before Running MAUDE

- Codeblock “make_example_df” provides a quick way to make a dataframe, to test the different functions of MAUDE
- Codeblock “basic_pipeline_comparision” allows comparison between data from my pipeline and johns pipeline.
- Codeblock “Data_exploration_violin” can create violin plots of the reads/umis and compare them to the median, or mean of the control sgRNAs
- Codeblock “check_data_for_maude” allows you to check the data before running MAUDE, to see if there is an imbalance in the amount of data between bins for example.

After Running MAUDE

- Codeblock “QQplot” contains functions for checking the significance of the results. It’s messy currently, but just load the function “plot_qq_maude” into the workspace, and then run plot_qq_maude(maude_gene_stats, “Experiment Name”, “AHSA1”)
  You can replace AHSA1 with another gene you wish to mark.

Advanced Options

- Read filtering can also be performed on data from Johns Cellranger & bcwithqc, using the “read_filter_john.sh” (single fastq file) or “read_filter_john_array.sh” (multiple fastq files).
    - Change input_dir to the main directory of johns output data.
    - Change output dir to your desired output directory
    - Change thresholds, to the path to the “thresholds.tsv” generated by “combined_pipeline_support.rmd”