Introduction

MEMDS summarizing scripts are a collection of scripts accompanying MEMDS analysis pipeline. The goal of the scripts is to aid the user with choosing optimal cut-offs for selecting consensus mutations and to summarize mutation consensus data. Additionally, they provide statistical information on read families and their associated barcodes for quality control purposes.

**Currently, are CLI (command line interface) only**, thus they need to be executed from the terminal of the operational system on which they are ran.

**Run requirements**

Operating system

The pipeline can run on any major Linux distributions and Mac systems with configured bash shell and installed R language. On Windows it is advised to use Windows Subsystem for Linux, since the scripts were configured and tested in Linux environment.

Distributed computing

The summarizing scripts are designed to run on distributed computing systems (clusters) managed by the SLURM Workload Manager. Running them on a different cluster system requires adjustment of job submission parameters within the wrapper scripts to its native commands.

To run the scripts on a local machine, job submission scripts can be converted into a standalone version by substituting SLURM job submission command (**“sbatch”**) into a direct call to the job script (see example below):

**Example #1:**

**Cluster submission:***sbatch --output="log.out" --error="log.err"\*

*-N1 -n1 -p hive1d,hive7d\*

*--wrap "****module load R/3.6.0-foss-2019a; Rscript analysis.R $input\_file $out\_file****"*

**Local run:**

***Rscript analysis.R $input\_file $out\_file***

***\*\*\*\*\****

**Note on the local run:** Some of the jobs can be quite resource intensive, so it is better to avoid running the pipeline on weaker machines. Additionally, since local machines doesn’t benefit from distributed computing capabilities, it is advisable to modify job submission scripts to launch only a single job at a time, by opening “for” loops implemented in them.

R

The summarizing scripts are written in R language and were tested with **R version 3.6.3**. If R is not installed on the system it can be downloaded from:   
***<https://cloud.r-project.org/>***

The scripts require the following R libraries to run properly:

- dplyr

- readr

- tidyr

- stringr

The libraries can be installed using the command *install.packages(“package”)* inside R environment.

**Preparation of the parameter files**MEMDS summarizing scripts utilize the same parameter files as MEMDS analysis pipeline during the run. See **“MEMDS\_pipeline\_guide”** for detailed information on preparation of these parameter files.

Additionally, summarizing scripts require **“summarize\_params.txt”** parameter table to run. The table should be placed inside **“config\_files”** directory under the **“scripts”** folder, together with the rest of the parameter files. **When filling the table**, make sure that filled-in data is not enclosed in quotation marks (e.g.: 40 **and not** “40”). This issue can occur sometimes, if the table is opened in Excel and then saved back as a text file.

**“Summarize\_params.txt”** table defines the following analysis parameters:

1) **ref\_name:** Name of the reference used to align studied reads. Should match the name of reference file inside the reference file directory specified by the MEMDS analysis pipeline. If multiple reference files are used, each name should be listed in a separate row.   
E.g: if reference file has a name “HBB.fa”, this field gets “HBB” value.

2) **ref\_seq:** Nucleotide sequence of the reference gene against which the reads are aligned. Can be copied form the corresponding reference file used by the MEMDS analysis pipeline.

3) **bases:** Nucleotide alphabet used for substitution counts by the scripts. By default it includes the common nucleotides - “A”, “C”, “T” and “G” and substitution counts include all possible combinations of them. The nucleotide alphabet is listed as a comma separated list - e.g.: A,C,T,G

4) **plasmid\_mut\_profile:** A combination of specific mutations that is used to distinguish spike-in plasmids used in the MEMDS experimental procedure from analyzed sequences. Listed as a comma separated list, with semi-colon separating profiles of different plasmids. E.g.: 39CT,46C-,47C-,52-TT;46CA,48T-,51-A,52GC.  
Since plasmid sequences are skipped during mutation count analyses, and used only for calculation of the enrichment factor, this field can also be used to filter out read families carrying specific mutation profiles. Any mutation profile specified here would be treated as a plasmid sequence and read families carrying it would be excluded from subsequent analyses.

5) **seq\_range:** A range of positions in the read-reference alignment in which mutations should be reported and analyzed. Counted relative to the first position of the alignment, which is specified as “1”. Listed in a format of <start\_pos>-<end\_pos>, e.g.: 31-70. Multiple ranges can be specified in a form of comma separated list.

6) **RS\_range:** A range of positions in the read-reference alignment specifying ROI (region of interest) - region of mutation enrichment. Counted relative to the first position of the alignment, which is specified as “1”. Listed in a format of <start\_pos>-<end\_pos>, e.g.: 65-70. Multiple ranges can be specified in a form of comma separated list.

7) **pos\_names:** A list of specific mutations to count, in addition to the total mutation counts per mutation type. The mutations are listed as a comma separated list, in a format: <mut\_pos><reference\_nucleotide><query\_nucleotide>. **“NONE” keyword** is used to specify that no specific mutations should be counted.  
E.g.: 39CT,44CA,49GA or NONE.

8) **coding\_start:** A variable used by the scripts to report mutation positions relative to the Translation Start Site (TSS) of the analyzed gene, in addition to its position on the read-reference alignment itself. Here two numbers, separated by comma, are specified (**e.g.: -30,1):**

a) **Position of the first alignment position relative to the TSS**. If TSS occurs before the analyzed portion of the gene, this value should represent position of the first read base in the analyzed gene (e.g. 1100). If TSS falls **inside** the read, this value should be negative and calculated as **1 - TSS position in the read**. E.g., if TSS occurs at **position 31 of the read**, it relative position is: **1 - 31 = -30.**

b) **Read orientation relative to the coding strand.** 1 - same orientation; -1 - opposite orientation (complimentary strand was sequenced).

Thus, **-30,1** example above means that the read and the coding strand of the gene are in the same orientation and TSS falls in the 31st position of the read.  
  
If TSS data is not relevant, **1,1** value should be specified to report same position for mutation location in the alignment and its location relative to the TSS.

9) **filtering\_pos:** Defines position of the control insertion used to identify artifacts occurring during the amplification stage. Reads with the control insertion are filtered out by the MEMDS analysis pipeline, but other mutations at this position can get reported. This parameter tells the pipeline to disregard them, since they do not represent mutations of interest. If control insertion is not used, position 0 should be specified.

10) **pos\_excluded:** Positions in the read-reference alignment that should be disregarded when performing mutation count and calculating error rate. The positions are counted relative to the first position of the alignment, which is specified as “1”. Multiple positions can be specified as a comma separate list. To consider **all** positions in the alignment, specify “0” here. E.g: 35,36,37.

11) **subs\_excluded:** Specific mutation types (e.g.: “CA”) that should be disregarded when performing mutation count and calculating error rate. Multiple mutation types can be specified as a comma separate list. **“NONE” keyword** is used to specify that all mutation types should be counted. E.g.: CA,GA or NONE.

12) **var\_excluded:** Specific mutations (e.g.: “39CT”) that should be disregarded when performing error rate calculation. These mutations are still included in the total count of mutations of this type, but also have their own count section. Thus they can be easily subtracted from total mutation count of the same type, if needed. Multiple mutations can be specified as a comma separate list. **“NONE” keyword** is used to specify that all mutation types should be counted. E.g.: 39CT or NONE.

**Selecting run parameters**When running the pipeline on a new data, it is advisable to use permissive parameters inside “Summarize\_params.txt” table for a first few samples - that is perform mutation counts and error rate calculations on all mutation types and all positions. After the initial data is obtained, it can be used to identify potential artifacts, “noisy” positions, etc. Then the calculations can be re-run with updated parameters to see their effect compared to the initial results.

**Quick start guide**This section provides basic information needed to run the summarizing scripts. For a more detailed information regarding output of each script refer to the next section. Unlike the MEMDS analysis pipeline, in which different steps need to be run in a sequential order, MEMDS summarizing scripts are largely independent from each other. Unless indicated otherwise, they can be run without any specific order.

Notes before the run

1) In the Quick start guide the “**$i”** symbol is used to indicate what numerical options should be provided after the command name, with available options listed in parentheses. **For example:** *bash my\_script.sh $i (1-3)* means that **my\_script.sh** accepts numerical options 1 to 3.   
If **no** numerical option is supplied, the script would print input and output parameters it uses and quit without submitting the job. This can be useful to validate that correct parameters are supplied to the wrapper script, without the need to run the whole task.

2) When running on a cluster, for each job submitted by the script the following message would appear: “Submitted batch job #job\_serial\_number”. **Always check that all running jobs were completed successfully, before downloading analysis results.** To check job status, use the following commands (for SLURM systems):

a) *squeue -u “username”| grep -c “username”* – this command displays number of jobs in queue for account defined by the “username”. Completed or canceled jobs would be removed from the queue.

b) *sacct -u “username”* – this command lists all the jobs that ran on the account defined by the “username” at current session. In the last column it indicates for each job if it is completed, canceled or still running. Ensure that all relevant jobs have ‘Completed’ status before submitting new ones!

3) **Job submission commands vary between different cluster systems**. Before the run remember to update the **“sbatch”** command in the pipeline job submission bash (“.sh”) files with the syntax relevant to your cluster.

4) **Remember to check the “.err” and the “.out” log files** produced during job run on the cluster. Log files are generated in the same folder as the result files produced by the pipeline in each step or sub-step. They record warnings and error messages raised by the script or by the cluster during job run.

5) Before running the pipeline, remember to put the folder containing pipeline scripts and associated parameter files into the relevant sample directory. If multiple samples are analyzed, “scripts” folder should be created in each sample’s directory.

Pipeline wrapper script

The wrapper script provides an interactive menu allowing users to navigate through the summarizing scripts and choose which analyses to execute.

To run the wrapper script, navigate from the terminal to the directory containing the helper script and use the command: *bash MEMDS\_summarizer\_wrapper.sh.* By default, the wrapper script is located in the directory of the summarizing scripts, but can be placed and run from any place on the machine. If scripts are placed on a remote machine, helper script should be placed and executed from the same machine, and not from the local host.

At the beginning of the run, helper script prompts to provide paths to the folders containing pipeline scripts, e.g: ***My\_computer/analysis/sample/scripts***.   
Multiple paths, pointing to script folders of different samples can be provided, separated by semi-colon, e.g:

***My\_computer/analysis/sample1/scripts;My\_computer/analysis/sample2/scripts****.*

After the paths are entered, the script offers on-screen menu with possible run options. Run options are organized into groups with each including one or more possible analysis options. If multiple options are available, additional menu would appear asking to choose specific analysis. If multiple paths are provided, choice prompt would appear separately for each path provided.

Running pipeline scripts directly

1) Navigate into the scripts folder from command line (***cd /path/to/sample/scripts***) to use the pipeline. All pipeline commands should be run from inside the scripts folder!

2) Choose the analysis to run from the options below:

**1) Barcode distribution analyses:**

1) **Analyze distribution of primary barcode families and secondary barcode groups within them:**

a) *bash BC\_dist.sh 1*

b) This script should be run **before** script 1-2.

2) **Calculate percent of families with more than 15 different 3’ barcode groups:**

a) *bash BC3\_above15.sh 1*

b) This script should be run **after** script 1-1.

3) **Check evenness of 3' barcode group sizes within read families:**

a) *bash BC\_eveness.sh 1*

4) **Check average number of distinct 3' barcode groups having more than one read within read families grouped by family size:**

a) *bash BC3\_size\_above1.sh 1*

5) **Report proportion of each of largest 3' barcode groups out of total family size averaged across read families of same size:**

a) *bash BC3\_freq.sh 1 $col\_report*

b) **User-defined parameters:**

*$col\_report* ***-*** Number of largest 3’ barcode groups to report in the output.

**2) Hamming distance analysis:**

1) **Analyze proportion of read families with primary barcodes found within Hamming distance of 1 from primary barcodes of larger families:**

a) *bash hamming\_dist\_parser.sh 1*

**3) Mutation count analyses:**

1) **Summarize consensus mutation counts per cut-off criteria set:**

a) *bash consensus-count\_summary.sh 1*

b) This script should be run **before** all other scripts in this section.

2) **Create a table of counted substitutions per cut-off criteria set:**

a) *bash substitution\_collate.sh 1 $genes\_analyzed $excl\_pos $excl\_sub $excl\_var*

b) **User-defined parameters:**

*$genes\_analyzed* ***-*** Names of the analyzed genes, should match the names in the **“summarize\_params.txt”** parametertable.Multiple genes can be supplied as a comma-separated list (e.g.: HBB,HBD,...).

*$excl\_pos* - Positions in the alignment to exclude when counting substitutions, comma-separated (e.g.: 39,52,...).

*$excl\_sub* - Substitution types to exclude from substitution counts, comma-separated (e.g.: CA,GA,...).

*$excl\_var* - Specific substitutions to exclude from substitution counts, comma-separated (e.g.: 39CT,47GA,...).

c) This script should be run **after** script 3-1.

3) **Create a table of expected and counted substitutions inside ROI per cut-off criteria set:**

a) *bash expected\_mut.sh 1 $genes\_analyzed $excl\_pos $excl\_sub $excl\_var*

b) **User-defined parameters:**

*$genes\_analyzed* ***-*** Names of the analyzed genes, should match the names in the **“summarize\_params.txt”** parameter table.Multiple genes can be supplied as a comma-separated list (e.g.: HBB,HBD,...).

*$excl\_pos* - Positions in the alignment to exclude when counting substitutions, comma-separated (e.g.: 39,52,...).

*$excl\_sub* - Substitution types to exclude from substitution counts, comma-separated (e.g.: CA,GA,...).

*$excl\_var* - Specific substitutions to exclude from substitution counts, comma-separated (e.g.: 39CT,47GA,...).

c) This script should be run **after** script 3-1.

4) **Create a table of error and mutation rates calculated per cut-off criteria set:**

a) *bash err\_rate\_collate.sh 1*

b) This script should be run **after** script 3-1.

5) **Recalculate error and mutation rates using custom parameters:**

a) *bash re\_calc\_error\_rate.sh 1 $genes\_analyzed $excl\_pos $excl\_sub $excl\_var*

b) **User-defined parameters:**

*$genes\_analyzed* ***-*** Names of the analyzed genes, should match the names in the **“summarize\_params.txt”** parameter table.Multiple genes can be supplied as a comma-separated list (e.g.: HBB,HBD,...)’

*$excl\_pos* - Positions in the alignment to exclude when counting substitutions, comma-separated (e.g.: 39,52,...).

*$excl\_sub* - Substitution types to exclude from substitution counts, comma-separated (e.g.: CA,GA,...).

*$excl\_var* - Specific substitutions to exclude from substitution counts, comma-separated (e.g.: 39CT,47GA,...).

c) This script should be run **after** script 3-1.

6) **Plot substitutions per alignment position, as a percent of all substitutions:**

a) *bash plot\_subs.sh 1 $genes\_analyzed $excl\_pos $excl\_sub $excl\_var*

b) **User-defined parameters:**

*$genes\_analyzed* ***-*** Names of the analyzed genes, should match the names in the **“summarize\_params.txt”** parameter table.Multiple genes can be supplied as a comma-separated list (e.g.: HBB,HBD,...).

*$excl\_pos* - Positions in the alignment to exclude when counting substitutions, comma-separated (e.g.: 39,52,...)

*$excl\_sub* - Substitution types to exclude from substitution counts, comma-separated (e.g.: CA,GA,...).

*$excl\_var* - Specific substitutions to exclude from substitution counts, comma-separated (e.g.: 39CT,47GA,...).

c) This script should be run **after** script 3-1.

7) **Create a table of plasmid to WT ratios per cut-off criteria set:**

a) *bash plasmid-WT\_ratio.sh 1 $genes\_analyzed $plasmid*

b**) User-defined parameters:**

*$genes\_analyzed* ***-*** Names of the analyzed genes, should match the names in the **“summarize\_params.txt”** parameter table.Multiple genes can be supplied as a comma-separated list (e.g.: HBB,HBD,...).

*$plasmid* - Plasmid sequence profiles distinguishing plasmids from the sequences of the analyzed gene(s). Should match one or more of the profiles appearing in the **“plasmid\_mut\_profile”** field of **“summarize\_params.txt”** parameter table. Multiple profiles should be separated by semi-colon, with individual positions of the same profile separated by commas, e.g.: **39CT,46C-,47C-;39CT,48T-,51-A**.

c) This script should be run **after** script 3-1.

**4) Primary barcode composition:**

1) **Check nucleotide frequencies within primary barcodes to assess barcode diversity:**

a) *bash BC5\_nucleotide\_dist.sh 1 $genes\_analyzed $id\_len*

b) **User-defined parameters:**

*$genes\_analyzed* ***-*** Names of the analyzed genes, should match the names in the **“summarize\_params.txt”** parameter table.Multiple genes can be supplied as a comma-separated list (e.g.: HBB,HBD,...).

*$id\_len* - Length of the barcode portion that includes sample identifier sequence that remains constant between all read families of the same sample.

**5) Relabeling incidence:**

1) **Check frequency of relabeling in the analyzed read families:**

a) *bash relabeling\_dist.sh 1 $genes\_analyzed $oligoD $id\_len*

b) **User-defined parameters:**

*$genes\_analyzed* ***-*** Names of the analyzed genes, should match the names in the **“summarize\_params.txt”** parameter table.Multiple genes can be supplied as a comma-separated list (e.g.: HBB,HBD,...).

*$oligoD* - Relabeling oligo sequence.

*$id\_len* - Length of the barcode portion that includes sample identifier sequence that remains constant between all read families of the same sample.

**Detailed pipeline guide**  
The section below provides detailed information regarding the summarizing scripts and their output. Unless noted otherwise, the scripts can be run independently from each other in no particular order.

Before the run  
1) Add summarizing scripts to the **“scripts”** directory containing MEMDS analysis pipeline scripts of the relevant sample.

2) Refer to the **“Preparation of the parameter files”** section above to organize pipeline parameter files needed for the analysis inside the ‘scripts’ folder.

3) Use the helper script, **“MEMDS\_summarizer\_wrapper.sh”,** situated in the “scripts” directory,and follow on-screen instructions to run the pipeline. Alternatively, enter the ‘scripts’ directory, to run pipeline scripts directly. Pipeline scripts need to be invoked only from inside the “scripts” directory to run properly.

4) Output generated by the summarizing scripts is stored in the same folder as the output of MEMDS analysis pipeline. A different output folder can be specified via “params\_1.sh” file (see “MEMDS\_pipeline\_guide” for a detailed explanation on its preparation). Remember that summarizing scripts utilize the same parameter files as the MEMDS analysis pipeline scripts, therefore changes in parameter files affect them as well.

5) In the job submission scripts notice the following parameters:

a) By default the scripts use **“BC3\_min”** parameter defined by the “params\_1.sh” file to run. This is to keep the scripts synchronized with the output of the MEMDS analysis pipeline. A different parameter can be assigned manually by adjusting “**$BC3\_min**”variable in the line “**for BC3\_min\_cutoff in $BC3\_min**” to a user specified number.

b) The scripts, by default, do not analyze the “others” files produced by the MEMDS analysis pipeline. To analyze these files, remove the following lines from the submission scripts:

***if [[ $r2 == \*"others"\* ]]; then***

***continue***

***fi***

c) As mentioned in the “MEMDS\_pipeline\_guide”, it is possible to analyze only reads having specific 3’ prime barcodes, by adjusting the parameter **“sam\_BX\_tag”** in the job submission file of mutation calling step. If a custom parameter was used for this variable in the MEMDS analysis pipeline, it should be adjusted accordingly in the job submission scripts here.

Log files

Running the scripts on a cluster system produces log files. They are stored under “logs” folder inside the results directory produced by the script. There are two types of files produced:

1) **“.err” file** - Captures error and warning messages produced by the scripts during the run. Before analyzing the output remember to check the“.err” files for presence of critical messages that might indicate failure of a given job (often accompanied by “FAILED” job status on the cluster). Additional, non-critical messages, commonly raised by the MEMDS summarizing scripts are:

- **Attaching package: ‘dplyr’**

**The following objects are masked from ‘package:stats’:**

**filter, lag**

**The following objects are masked from ‘package:base’:**

**intersect, setdiff, setequal, union**

This message informs the user that functions of “dplyr” library are used instead of same-name functions in other libraries of R, and is displayed each time the “dplyr” package is used.

- **Warning message: Grouping rowwise data frame strips rowwise nature** - Informs the user that grouping data in the table would cause it to be analyzed by defined groups and not by each row separately.

- **In dir.create(outdir, recursive = TRUE) : Path already exists** - Informs the user that they attempt to create a directory that already exists.

These messages are raised as a part of a normal run of the scripts, and can be disregarded when looking at “.err” files.

- **Parsed with column specification: cols(X, Y, Z)** - Some functions reading the input data in tabular format print a message listing data type in each column of the input table (numeric integer, numeric double, character, etc.). This can be used for diagnostic purposes, to see that input data being read correctly by the script and that input table columns do not include unexpected characters (e.g. column that supposed to contain only numbers is treated as alphanumeric data).

2) **“.out” file** - Captures on-screen information printed by the scripts. “.out” file format is standardized across summarizing scripts and contains the following information:

- First line of the file indicates when the script run was started:

**Starting Mon Jan 1 12:00:00 2022**

- Second line contains statement **“Run args”**, indicating that several lines below it contain information on the run parameters utilized by the script: names of the input and output files, information on substitution counts that are excluded from the analysis, etc. The number of argument lines differs between summarizing scripts, depending on the number of the input parameters they need for their run.

- After the parameter lines appears the following line: “**Detected N input files in folder**”, whereby “N” is the number of input files detected and processed by the script and “folder” refers to the input folder in which these files are found. After this line all processed files are listed in a format: **“1 / N : Name of the first file”**, **“2 / N : Name of the second file”**, with each processed file listed in a separate line. This part is optional and appears only when multiple files are processed by the script.

- The second from last line, appearing after the list of processed files (or “Run args” lines), states the total run time of the script, in seconds: “**X secs**”.

- The last line indicates time of run completion and contains keyword “Done” to indicate that the script executed all the code successfully: “**Done** **Mon Jan 1 12:01:00 2022**” .

The information in the “.out” files is for diagnostic purposes. It allows the user to validate that correct parameters were used for the analysis, that all expected files were processed by the script and assess how long it might take to run the analysis on their machine for the next samples.

The most important line is the last one. Cluster machines can sometimes terminate jobs mid-run, without issuing any error messages. Therefore it is important to see the “Done” line in the log files indicating that all the code of the script was executed.

**Barcode distribution analyses**

**1) Analysis of primary and secondary barcode groups’ distribution  
Step description:**This script groups barcode families (groups of reads sharing the same primary - 5’ - barcode) by their size and reports how many families of given size are found within the sample. Next, for each group of same-sized families the script reports how many different secondary barcode (3’) groups are associated with them (mean, median and maximal values). Additionally, it reports what proportion of read families in each size group has more than 15 different secondary barcode groups associated with them.

Distribution data provided by the script can help to choose family size and secondary barcode group number cut-off criteria for reporting consensus mutations generated by the MEMDS analysis pipeline.

The information on presence of more than 15 different secondary barcode groups is used to assess secondary barcode inflation in the analyzed sample, relative to the family size. In the MEMDS protocol used by us linear amplification step generated up to 15 copies of the target molecule. Therefore read families were expected to have at most 15 different secondary barcodes associated with them, assuming that each analyzed molecule gets a unique primary barcode before linear amplification step. In practice, plotting fraction of families with more than 15 secondary barcode groups vs family size created a sigmoidal-like curve, with almost all families of larger sizes having more than 15 secondary barcode groups associated with them. And this pattern repeated itself across analyzed samples.

If a custom MEMDS procedure is used, the number of secondary barcodes indicating inflation might be different from 15. To adjust the analysis accordingly, in the script **“BC\_dist.R”** adjust the expression “**total\_BC3\_types > 15”** in line 53 to a relevant number.

**Run instructions:**a) Check that the **“factors\_table.txt”** contains correct values in the columns: “sort\_refs” and “sort\_ref”. In case of incorrect values - refer to the “MEMDS\_pipeline\_guide” regarding parameter file preparation.

b) **Run script:** *bash BC\_dist.sh 1*

c) Wait for the job to end. Use the “sacct” and the “squeue” commands on the cluster to check job status.

d) Inspect the output files in the “tables.BC3cutoff$BC3\_min\_cutoff.BC\_dist” directory, whereby “BC3\_min\_cutoff” is defined by the “BC3\_min” parameter in the “params\_1.sh” file.

**Output description:**

This step analyzes distribution of read family sizes and distribution of secondary barcode group counts across read families. Upon its completion following files are added to the output directory:

1) **<sample\_name>.<ref\_name>.bwa.sorted.bam.BC\_log.err,**

**<sample\_name>.<ref\_name>.bwa.sorted.bam.BC\_log.out** - Log files produced during the run of the job on the cluster. Added to the “logs” folder inside output directory. Check that the “.err” file doesn’t contain error messages and that the “.out” file contains correct data for run parameters and indicates that the job was completed successfully.

2) **<sample\_name>.<ref\_name>.bwa.sorted.bam.BC\_dist.txt** - Tab-delimited summary table outlining analysis results. Contains the following columns:

a) **Family\_size** - number of reads in the read family.

b) **Family\_count** - number of read families of given size.

c) **Mean\_BC3\_types** - average number of secondary barcode groups per read family, calculated across same-sized read-families.

d) **Median\_BC3\_types** - median number of secondary barcode groups per read family, calculated across same-sized read-families.

e) **SD\_BC3\_types** - standard deviation in the number of secondary barcode groups associated with the same-sized read-families.

f) **Max\_BC3\_types** - the largest number of secondary barcode groups associated with any of the read families of given size.

g) **BC3\_above15\_freq** - proportion of read families of given size that have more than 15 different secondary barcode groups associated with them.

**2) Calculate percent of families with more than 15 different 3’ barcode groups  
Step description:**This script takes information generated by the “**BC\_dist**”script and outputs frequency of read families with more than 15 different secondary barcodes out of all families with size above 15 reads in the sample. This makes it easier to assess overall tendency of barcode inflation in the sample, while the analysis above is useful to look at the relationship between family size and barcode inflation.

If a custom MEMDS procedure is used, the number of secondary barcodes indicating inflation might be different from 15. To adjust the analysis accordingly, in the script **“BC3\_above15.R”** adjust the expression “**tab$Family\_size > 15”** in line 41 to a relevant number.

**Run instructions:**a) Check that the “**BC\_dist.sh**”script was run before running this analysis.

b) **Run script:** *bash BC3\_above15.sh 1*

c) Wait for the job to end. Use the “sacct” and the “squeue” commands on the cluster to check job status.

d) Inspect the output in the “tables.BC3cutoff$BC3\_min\_cutoff.BC\_dist” directory, whereby “BC3\_min\_cutoff” is defined by the “BC3\_min” parameter in the “params\_1.sh” file.

**Output description:**

This step analyzes frequency of read families with more than 15 different secondary barcodes associated with them per analyzed sample. Upon its completion following files are added to the output directory:

1) **<sample\_name>.BC3\_above15\_log.err, <sample\_name>.BC3\_above15\_log.out** - Log files produced during the run of the job on the cluster. Added to the “logs” folder inside output directory. Check that the “.err” file doesn’t contain error messages and that the “.out” file contains correct data for run parameters and indicates that the job was completed successfully.

2) **<sample\_name>.BC3\_above15.txt** - Tab-delimited summary table outlining analysis results. Contains the following columns:

a) **Sample** - names of the analyzed files, matches names of the output files produced by the “BC\_dist” script.

b) **Families\_15+\_reads** - number of read families in the sample that have more than 15 reads and thus can have more than 15 different 3’ barcode groups associated with them.

c) **BC3\_types\_above15** - frequency of read families that have more than 15 different secondary barcode groups associated with them among read families with size 16 reads or more.

**3) Check evenness of 3' barcode group sizes within read families  
Step description:**This script uses **Shannon Evenness Index (SEI)** to assess diversity in sizes of secondary barcode groups associated with read family. For more information on the calculation of evenness see: [Shannon Evenness Index](https://www.statology.org/shannon-diversity-index/). For the purpose of this analysis the important thing to take away is that SEI can have a value between 0 and 1, with 1 being complete evenness in sizes of compared groups.

The SEI can be used as another way of looking at secondary barcode inflation issue. After linear amplification step, during which the 3’ barcode is attached, the molecules undergo PCR amplification steps. Since each linearly amplified copy of the original molecule is further amplified by PCR independently of others, one might expect to see diversity in sizes of secondary barcode groups within the read family. Of course, if the total size of the read family is small, there wouldn’t be much place for size diversity of secondary barcode groups, but in sufficiently large families it should become apparent.

In our samples, across different analyzed genes, we indeed observed decrease in SEI, indicating larger diversity of 3’ barcode group sizes, with increase of the total size of read families, but only up to a point. Afterwards SEI values started to increase back. Looking at these families in more detail, we saw that they tend to have a fairly large number of 3’ barcode groups having just one read in them, with overall number of 3’ barcode groups often being above 15. Overall, it seems that increase in SEI of larger families has correlation to secondary barcode inflation in these families, likely caused by these single-read groups.

**Run instructions:**a) Check that the **“factors\_table.txt”** contains correct values in the columns: “sort\_refs” and “sort\_ref”. In case of incorrect values - refer to the “MEMDS\_pipeline\_guide” regarding parameter file preparation.

b) **Run script:** *bash BC\_eveness.sh 1*

c) Wait for the job to end. Use the “sacct” and the “squeue” commands on the cluster to check job status.

d) Inspect the output in the “tables.BC3cutoff$BC3\_min\_cutoff.BC\_evenness” directory, whereby “BC3\_min\_cutoff” is defined by the “BC3\_min” parameter in the “params\_1.sh” file.

**Output description:**

This step analyzes size diversity of secondary barcode groups associated with read families. Upon its completion following files are added to the output directory:

1) **<sample\_name>.<ref\_name>.bwa.sorted.bam.BC\_evenness.err,**

**<sample\_name>.<ref\_name>.bwa.sorted.bam.BC\_evenness.out** - Log files produced during the run of the job on the cluster. Added to the “logs” folder inside output directory. Check that the “.err” file doesn’t contain error messages and that the “.out” file contains correct data for run parameters and indicates that the job was completed successfully.

2) **<sample\_name>.<ref\_name>.bwa.sorted.bam.BC\_evenness.txt** - Tab-delimited summary table outlining analysis results. Contains the following columns:

a) **Family\_size** - number of reads in the read family.

b) **Family\_count** - number of read families of given size.

c) **Mean\_evenness** - average SEI value of secondary barcode group sizes across across same-sized read-families..

d) **Median\_evenness** - median SEI value of secondary barcode group sizes across same-sized read-families..

e) **SD\_evenness** - standard deviation of the SEI values of secondary barcode group sizes across same-sized read-families.

f) **Min\_evenness** - the smallest SEI value obtained for any read family among read families of a given size.

**4) Check number of secondary barcode groups with more than one read in them  
Step description:**This script calculates what proportion of secondary barcode groups in read family has more than one read in them and what fraction of reads, out of total reads in the family, is included in these groups. The results are summarized across families of same size.

When looking at the issue of secondary barcode inflation, it is likely to assume that artifact groups would be mostly of small size. To create larger artifact groups, the issue would need to happen before the PCR step, which is less likely if the primary barcode sequences used in the experiment are diverse enough. Therefore, by looking at amount of 3’ barcode groups in read families we can assess how much data belongs to the groups that are more likely to be true.

In our data we noticed that usually the amount of secondary barcode groups having more than one read in them tends to be, on average, less than half of all secondary barcode groups in read families of given size. At the same time, these multi-read secondary barcode groups contain the majority of reads in the said read families. Overall, this indicates that even if some read families have secondary barcode inflation, the majority of reads in them tends to belong to the larger 3’ barcode groups and not the small ones. This increases the confidence that any mutation appearing at sufficiently high frequency within these read families is likely to be linked to the more trustworthy secondary barcode groups.

To customize the minimal size of secondary barcode groups counted by the script as large groups, adjust the the expression “**count\_list > 1”** in lines 29 and 36 of the **“BC3\_size\_above1.R”** script.

**Run instructions:**a) Check that the **“factors\_table.txt”** contains correct values in the columns: “sort\_refs” and “sort\_ref”. In case of incorrect values - refer to the “MEMDS\_pipeline\_guide” regarding parameter file preparation.

b) **Run script:** *bash BC3\_size\_above1.sh 1*

c) Wait for the job to end. Use the “sacct” and the “squeue” commands on the cluster to check job status.

d) Inspect the output in the “tables.BC3cutoff$BC3\_min\_cutoff.BC3\_above1” directory, whereby “BC3\_min\_cutoff” is defined by the “BC3\_min” parameter in the “params\_1.sh” file.

**Output description:**

This step analyzes size diversity of secondary barcode groups associated with read families. Upon its completion following files are added to the output directory:

1) **<sample\_name>.<ref\_name>.bwa.sorted.bam.BC3\_above.err,**

**<sample\_name>.<ref\_name>.bwa.sorted.bam.BC3\_above.out** - Log files produced during the run of the job on the cluster. Added to the “logs” folder inside output directory. Check that the “.err” file doesn’t contain error messages and that the “.out” file contains correct data for run parameters and indicates that the job was completed successfully.

2) **<sample\_name>.<ref\_name>.bwa.sorted.bam.BC3\_above1.txt** - Tab-delimited summary table outlining analysis results. Contains the following columns:

a) **Family\_size** - number of reads in the read family. Ordered from largest to smallest family.

b) **Family\_count** - number of read families of given size.

c) **Group counts** -Three columns following the “Family\_count” column show mean, median and standard deviation in the number of 3’ barcode groups with more than one read in them across read families of given size (“**Mean\_types\_above1**”, “**Median\_types\_above1**”, “**SD\_types\_above1**”).

d) **Read counts** - The columns following the group count columns show mean, median and standard deviation in the fraction of reads included in the 3’ barcode groups with more than one read in them, out of total number of reads in the same-sized read families (“**Mean\_reads\_above1**“, “**Median\_reads\_above1**“, “**SD\_reads\_above1**“).

**5) Report proportion of largest 3' barcode groups out of total read family size  
Step description:**This script organizes secondary barcode groups within read families by their size and calculates proportion of N largest groups, out of total family size, with “N” being defined by the user. The proportions are calculated per read family and then averaged across same-sized read families.

The output of this script is used to create publication figures depicting secondary barcode group distribution and variety within analyzed read families. The goal is to show that read families are not over-dominated by reads coming from a single linearly amplified copy of the original molecule, which negates the benefits of having two barcodes attached to the analyzed molecules.

As discussed above, in larger families small-sized, especially single-read, secondary barcode groups might represent artifacts of 3’ barcode inflation. To account for this, the script produces two types of output. One reports data on secondary barcode groups of all sizes. The second analyzes only groups having more than one read in them. In the latter case the total number of reads in the family used for proportion calculation is adjusted to total number of reads in secondary barcdode groups having more than one read (called “Effective family size”).

The report on secondary barcode groups with more than one read can be customized to a user-defined minimal group size threshold by adjusting the script **“BC3\_freq.R”**. To do so, change the expressions “**count\_list > 1”** in line 48 and **“BC3\_above1(barcode3\_all\_counts, 1)”** in line 166 to a required number.

**Run instructions:**a) Check that the **“factors\_table.txt”** contains correct values in the columns: “sort\_refs” and “sort\_ref”. In case of incorrect values - refer to the “MEMDS\_pipeline\_guide” regarding parameter file preparation.

b) **Run script:** *bash BC3\_freq.sh 1 $col\_report*

**User-defined parameters:**

*$col\_report* ***-*** Number of largest 3’ barcode groups to report in the output.

c) Wait for the job to end. Use the “sacct” and the “squeue” commands on the cluster to check job status.

d) Inspect the output in the “tables.BC3cutoff$BC3\_min\_cutoff.BC3\_freq” directory, whereby “BC3\_min\_cutoff” is defined by the “BC3\_min” parameter in the “params\_1.sh” file.

**Output description:**

This step outputs proportions of “N” largest 3’ barcode groups averaged across same-sized read families, with “N” defined by the user. Upon its completion following files are added to the output directory:

1) **<sample\_name>.<ref\_name>.bwa.sorted.bam.BC3\_freq.err,**

**<sample\_name>.<ref\_name>.bwa.sorted.bam.BC3\_freq.out** - Log files produced during the run of the job on the cluster. Added to the “logs” folder inside output directory. Check that the “.err” file doesn’t contain error messages and that the “.out” file contains correct data for run parameters and indicates that the job was completed successfully.

2) **<sample\_name>.<ref\_name>.bwa.sorted.bam.BC3\_freq.txt** - Tab-delimited summary table outlining secondary barcode group proportions. Contains the following columns:

a) **Family\_size** - number of reads in the read family. Ordered from largest to smallest family.

b) **Family\_count** - number of read families of given size.

c) **Secondary** **barcode group frequencies and SD** - the “N” columns, following “Family\_count” column, report proportion of the largest 3’ barcode groups in the read families out of total family size. The number of reported groups is defined by the user. Frequencies are calculated for each read family separately and then averaged across read families of same size. The columns are ordered in descending order, meaning that first column represents average proportion of the largest secondary barcode group, the second - second largest group, and so on, till “N” groups are reported. Alongside each average proportion reported its standard deviation across read families of given size. If families of certain size have less different 3’ barcode groups than user-defined “N”, secondary barcode group number is completed to “N” by zeroes.

3) **<sample\_name>.<ref\_name>.bwa.sorted.bam.BC3\_above1\_freq.txt** - Tab-delimited summary table outlining proportions of secondary barcode groups with more than one read in them. Its purpose is to show that even if single-read secondary barcode groups are discarded, the remaining reads in the family are not over-dominated by reads coming from a single linearly amplified copy of the original molecule. The table has the same format as the “**BC3\_freq”** table above, with addition of two columns:

a) **Effective\_size\_mean** - effective size of read families averaged over read families of the same size. Effective size represents total number of reads remaining in the read families after exclusion of single-read secondary barcode groups.

b) **Effective\_size\_SD** - standard deviation in the effective size of read families having same size when all secondary barcode groups are counted.

4) **<sample\_name>.<ref\_name>.bwa.sorted.bam.top\_5000.txt** - Tab-delimited summary table outlining secondary barcode information for 5000 largest families in the analyzed sample. This file allows the user to concentrate on the largest read families in the sample and perform on them their own secondary barcode distribution analyses. Contains the following columns:

a) **ID** -a unique primary barcode sequence serving as an identifier of a read family.

b) **Family\_size** - total number of reads in the read family.

c) **Total\_BC3\_types** - count of different secondary barcodes associated with the read family.

d) **BC3\_counts\_ordered** - number of reads in each secondary barcode group of the read family, ordered from largest to smallest.

e) **Effective\_size** - number of reads in the read family after exclusion of reads contained in the single-read secondary barcode groups.

f) **Effective\_types** - count of different secondary barcodes associated with the read family, not counting single-read secondary barcode groups.

g) **BC3\_counts\_above1** - number of reads in each secondary barcode group of the read family, ordered from largest to smallest, not counting single-read secondary barcode groups.

**Hamming distance analysis**

**1) Analysis of proportion of read families with primary barcodes found within Hamming distance of 1 from primary barcodes of larger families  
Step description:**This script analyzes primary barcode sequences of same-size read families and calculates what fraction of families of given size have sequences with Hamming distance of one from barcode sequences of larger read families. It also outputs the counted-to-expected ratio of fractions to test for enrichment of such barcodes in read families of given size.

The MEMDS method uses unique primary barcodes to distinguish between reads originating from unique molecules in the analyzed sample, which allows it to capture ultra-rare variants with high degree of accuracy. However during amplification process of the reads random mutations might get introduced into barcode sequences, causing some reads to appear as a unique family when in fact they belong to a larger family. These mutations are not expected to be frequent, given that the variable part of primary barcode sequence is only 14 bp long. Thus such artifacts are not likely to have more than a single difference in their primary barcode sequence from primary barcode sequences of larger families. Now, among randomly generated sequences used for barcoding purposes some of them might be similar in all, but one position, to begin with, therefore not all read families found by this analysis represent artifacts. However, if sufficiently large number of such barcodes is found among read families of certain size, relative to the expected number, this might indicate an issue.

To measure the differences between primary barcode sequences of read families of a given size and larger read families, we used Hamming distance metric. The Hamming distance indicates the number of different digits or letters in a pair of strings of same length. To read more about Hamming distance calculation refer here: **[Hamming distance](https://www.omnicalculator.com/other/hamming-distance)**.   
The expected number of primary barcode sequences found within Hamming distance of 1 from primary barcodes of larger families is given by:

**Whereby:**

*nA* - number of different primary barcode sequences among read families of a given size.

*nB* - number of different primary barcode sequences among read families larger than a given size.

*seq\_length* ***-*** length of the compared sequences.

***The formula is provided by courtesy of Dr. Yuval Nov, Department of Statistics,  
University of Haifa, Israel***

In the MEMDS protocol we used primary barcode sequences had **14 varying bases** serving as a unique identifier of each analyzed molecule and **4 constant bases** serving as a sample ID shared between all molecules of the same sample. Accordingly, in the expected count formula length of the compared sequences was set to 14 bps, since only these 14 bases could contribute to differences between primary barcodes. If different primary barcode setup is used, in the script **“hamming\_dist\_parser.R”** adjust the variable **“nchar(v1)[1] - 4”** in line 84 to the required number of constant sample identifying bases.

In our analyses we noticed that the counted-to-expected ratio of primary barcodes within Hamming distance of 1 from primary barcodes in larger families usually remained higher than one, sometimes substantially so. As mentioned above, there can be legitimate primary barcode sequences differing by only one position from other barcode sequences and that can affect calculated ratios. However some not easily quantifiable biases might also exist both in the process of primary barcode sequence generation and in their propensity to get attached to the analyzed molecules. Since the formula for calculating expected frequency is based on an assumption of total randomness of these processes, such biases can create imprecision in the statistical calculation. Therefore, we opted to look at the behavior of the counted-to-expected ratios as a function of analyzed read families’ size and sought family size cut-off above which the ratio would remain mostly stable. Random sequencing artifacts are not expected to produces systematic and constant influences, therefore reaching a point of stability likely indicates that these artifacts stop being a major influence.

We found out that the disbalance between counted and expected read families usually was the highest for read families having size of one read and sharply decreased when family size increased to two reads. Further decrease, albeit smaller, was observed between sizes of two and three reads. After that the counted-to-expected ratio remained fairly stable and fluctuated around similar numbers for a wide range of family sizes. Importantly, we saw this effect across different analyzed genes, suggesting that it is not a gene-specific phenomena.

**Run instructions:**a) Check that the **“factors\_table.txt”** contains correct values in the columns: “sort\_refs” and “sort\_ref”. In case of incorrect values - refer to the “MEMDS\_pipeline\_guide” regarding parameter file preparation.

b) **Run script:** *bash hamming\_dist\_parser.sh 1*

c) Wait for the job to end. Use the “sacct” and the “squeue” commands on the cluster to check job status.

d) Inspect the output in the “tables\_consensus.BC3cutoff$BC3\_min\_cutoff.hamming” directory, whereby “BC3\_min\_cutoff” is defined by the “BC3\_min” parameter in the “params\_1.sh” file.

**Output description:**

This step analyzes primary and secondary barcode read groups’ distribution. Upon its completion following files are added to the output directory:

1) **<sample\_name>.<ref\_name>.bwa.sorted.bam.hmm.err,**

**<sample\_name>.<ref\_name>.bwa.sorted.bam.hmm.out** - Log files produced during the run of the job on the cluster. Added to the “logs” folder inside output directory. Check that the “.err” file doesn’t contain error messages and that the “.out” file contains correct data for run parameters and indicates that the job was completed successfully.

2) **<sample\_name>.<ref\_name>.bwa.sorted.bam.hamming.txt** - Tab-delimited summary table outlining analysis results. Contains the following columns:

a) **Family\_size** - number of reads in the read family.

b) **Family\_num** - number of read families of given size.

c) **Families\_gt\_size** - number of read families above given size.

d) **Hamming\_1** - count of read families of given size having primary barcode sequences within Hamming distance of 1 from primary barcode sequences belonging to larger size families.

e) **Hamming\_gt1** - count of read families of given size having primary barcode sequence within Hamming distance greater than 1 from primary barcode sequences belonging to larger size families.

f) **Hamming\_1\_fract** - fraction of read families, from all families of given size, having primary barcode sequences within Hamming distance of 1 from primary barcode sequences belonging to larger size families.

g) **Hamming\_gt1\_fract** -fraction of read families, from all families of given size, having primary barcode sequences within Hamming distance greater than 1 from primary barcode sequences belonging to larger size families.

h) **Hamming\_1\_exp\_fract** - expected fraction of read families, from all families of given size, having primary barcode sequences within Hamming distance of 1 from primary barcode sequences belonging to larger size families. Calculated by the formula outlined above.

g) **Obs\_to\_exp** - ratio of counted to expected fractions of read families having primary barcode sequences within Hamming distance of 1 from primary barcode sequences belonging to larger size families. Calculated as **Hamming\_1\_fract/Hamming\_1\_exp\_fract**.

**Mutation count analyses**

**1) Summarization of consensus mutation counts per cut-off criteria set  
Step description:**This scripttakes consensus mutation tables generated by the MEMDS analysis pipeline and summarizes their data in a convenient, human readable form. **Important:** this script should be run **before** any other scripts in this section, since they utilize its output during their run.

**Run instructions:**a) Check that the **“factors\_table.txt”** contains correct values in the columns: “sort\_refs” and “sort\_ref”. In case of incorrect values - refer to the “MEMDS\_pipeline\_guide” regarding parameter file preparation. Check that **“summarize\_params.txt”** parameter table contains correct data and is placed in the “config\_files” folder, as described above.

b) **Run script:** *bash consensus-count\_summary.sh 1*

c) Wait for the job to end. Use the “sacct” and the “squeue” commands on the cluster to check job status.

d) Inspect the output in the “tables\_consensus.BC3cutoff$BC3\_min\_cutoff.summary” directory, whereby “BC3\_min\_cutoff” is defined by the “BC3\_min” parameter in the “params\_1.sh” file.   
Under the main directory, separate folder is generated for each analyzed gene and each treatment (control or experiment), having a format: “**<sample\_name>.<ref\_name>.bwa.sorted.bam.cutoffs-update**”. Under it sub-folders are generated, to store the output, for each set of combined cutoff criteria, e.g.: “**<sample\_name>.<ref\_name>.mutFreq0.0\_readCount0\_BC3WithMut0\_  
BC3above0**”. Sub-folder name includes the following cut-off values:

a) **mutFreq<W>** - mutation frequency threshold.

b) **readCount<X>** - read family size threshold.

c) **BC3WithMut<Y>** - threshold of different 3’ barcode groups associated with the allele within the read family.

d) **BC3above<Z>** - threshold of different 3’ barcode groups associated with the allele within the read family that have above user-specified number of reads in each group.

**Output description:**

This step summarizes consensus mutation tables. Upon its completion following files are added to the output directory:

1) **<sample\_name>.<ref\_name>.bwa.sorted.bam.cutoffs-update.err,**

**<sample\_name>.<ref\_name>.bwa.sorted.bam.cutoffs-update.out** - Log files produced during the run of the job on the cluster. Added to the “logs” folder inside output directory. Check that the “.err” file doesn’t contain error messages and that the “.out” file contains correct data for run parameters and indicates that the job was completed successfully.

2) **Mut\_totals\_summary.txt** - A summary file of consensus mutation data. Separate summary is generated for each combined cut-off criteria set. The data has a tab-delimited format, with first column containing data description and next columns containing the summarized counts. Summary file contains the following sections:

a) **Total\_family\_counts** - this section contains counts of total read families that pass the combined cut-off criteria, WT families, families containing ambiguous positions (where not mutation nor WT allele pass the cut-off criteria) and plasmid counts. Plasmid mutation signatures are defined by the “**plasmid\_mut\_profile**” field in the **“summarize\_params.txt”** parameter table, as described in the **“Preparation of the parameter files”** section above. Besides each plasmid signature appear **two** numbers: first one indicates number of read families containing plasmid signature with one mismatch allowed; the second one - number of read families containing plasmid signature having exact match to given mutation signature. If mutation profile of the plasmid contains less than five different identifying mutations, only exact matches would be counted in both cases.   
To customize plasmid matching parameters, in the file **“consensus-count\_summary\_v3.R”** adjust line 205:

**match\_size = ifelse(length(plasmid\_list[[k]]) > 4, (length(plasmid\_list[[k]]) - 1), length(plasmid\_list[[k]]))**

Here, the expression “**length(plasmid\_list[[k]]) > 4**” is the condition defining how many mutation should mutation profile contain to allow mismatches; the expression “ **(length(plasmid\_list[[k]]) - 1)**” defines how many mismatches to allow; and the expression “**length(plasmid\_list[[k]])**” defines that no mismatches are allowed if the condition “**length(plasmid\_list[[k]]) > 4**” is not satisfied.

b) **Special\_position\_counts** - Counts of specific substitutions defined by the “**pos\_names**” field in the **“summarize\_params.txt”** parameter table. See the **“Preparation of the parameter files”** section above for detailed description of the parameter table.

c) **Excluded\_variant\_counts** - Counts of specific substitutions defined by the “**var\_excluded**” field in the **“summarize\_params.txt”** parameter table. See the **“Preparation of the parameter files”** section above for detailed description of the parameter table. Separate counts of these substitutions are provided, to make it easier removing them from total substitution counts of the same type.

d) **Counts\_per\_substitution** - Counts of substitutions inside the read, separated by type (CA count, GA count, etc.). Read families containing ambiguous positions and read families containing plasmid mutation signature (as defined by the “**plasmid\_mut\_profile**” field in the **“summarize\_params.txt”** parameter table) are excluded from the counts. Substitution types that are counted are defined by the nucleotide alphabet listed in the **“bases”** field of the **“summarize\_params.txt”** file.

e) **Counts\_per\_substitution\_RS** - Counts of substitutions inside the ROI - region of interest portion of the read that undergoes mutation amplification step during the MEMDS procedure. Separated by type (CA count, GA count, etc.). The counts are provided for all positions inside ROI and excluding positions specified by the “**pos\_excluded**” field in the **“summarize\_params.txt”** parameter table.  
Read families containing ambiguous positions and read families containing plasmid mutation signature (as defined by the “**plasmid\_mut\_profile**” field in the **“summarize\_params.txt”** parameter table) are excluded from the counts. Substitution types that are counted are defined by the nucleotide alphabet listed in the **“bases”** field of the **“summarize\_params.txt”** file.

f) **Counts\_per\_substitution\_outside\_RS** - Counts of substitutions in the read portion flanking the ROI region. Separated by type (CA count, GA count, etc.). The counts are provided for all positions inside flanking regions and excluding positions specified by the “**pos\_excluded**” field in the **“summarize\_params.txt”** parameter table.  
Read families containing ambiguous positions and read families containing plasmid mutation signature (as defined by the “**plasmid\_mut\_profile**” field in the **“summarize\_params.txt”** parameter table) are excluded from the counts. Substitution types that are counted are defined by the nucleotide alphabet listed in the **“bases”** field of the **“summarize\_params.txt”** file.

g) **Total\_substitution\_counts** - total substitution counts in the analyzed read families, along the whole read, inside ROI region and in ROI flanks. Contains counts of all substitutions and counts excluding substitution types and positions specified by the “**subs\_excluded**” and “**pos\_excluded**” fields in the **“summarize\_params.txt”** parameter table.

h) **Error+mut rate** - this field specifies error and corrected mutation rates calculated from substitution count data outlined above.

**Error rate** represents substitution counts in the ROI-flanking regions normalized by sequence length and read family number. The MEMDS method is designed to search for ultra-rare variants, hence it is unlikely to detect mutations outside ROI region that undergoes enrichment specifically for this purpose. Therefore mutations found outside ROI are likely to represent artifacts and can be used for assessing error rate of the method. While this assumption is somewhat restrictive, since some of these mutations might represent true variants, it is better to err on the caution side when assessing error rate of the method.

**Error rate** is calculated as follows:

**Whereby:**

*- substitution\_count\_flanks* - count of substitutions in read regions flanking the ROI, while **excluding** specific positions and substitution types defined by the “**pos\_excluded**”,”**subs\_excluded**” and ”**var\_excluded**” fields defined in the **“summarize\_params.txt”** parameter table.

*- flank\_length* - number of base pairs in the flanking regions used for substitution count above. If specific positions are excluded from the counts, flank\_length is adjusted accordingly

*- family\_number* ***-*** count of “good” read families (not ambiguous, plasmids, etc.).

**Error rate** is calculated using three types of read family counts: WT families only; WT and substitution containing families (no families containing indels); and WT, substitution and indel containing families. The users can choose then which type of error rate calculation to use in their analyses.

**Mutation rate** is calculated similarly to the error rate, but using substitution counts **inside** ROI and not in its flanking regions. Accordingly it is normalized by ROI length:

**Mutation rate** represents rate of mutations captured inside ROI, before correction by enrichment factor. **Corrected mutation rate**, shown in the summary file, is mutation rate minus error rate. It is used to show that even before correction by the enrichment factor, higher number of substitutions per base is observed inside ROI compared to the flanking regions, as expected.

Additional correction that can be applied to either **error or mutation rates** is correction to **sequence length** by which substitution counts are normalized due to removal of specific substitution types from consideration. In each base of a sequence potentially can occur 3 different types of substitutions, therefore each base contributes 3/3 potential changes. However, if certain substitution types are removed from consideration, for example CA, now all ‘C’ bases can contribute only 2/3 changes, since third change is not considered. Thus ‘C’ is counted as 2/3 of a base, and then all bases in the sequence are counted in such a manner, **effective length** of the sequence is obtained. For example ‘CACA’ sequence has a length of 4 bps. But if ‘CA’ change is not counted, its **effective length** than can contribute to potential changes is 3⅓, since each ‘C’ contributes 2/3 changes. To make this correction, multiply error or mutation rate by relevant sequence length and divide the result by the calculated effective length. This correction is not applied automatically, since not all works use it.

i) **Indel\_counts** - this section lists how many read families, excluding rejected, ambiguous and plasmids, contain indels of a given type. Each indel type (e.g. 82CAG- , 84-ACA) is listed in a separate row. The indels are listed in the order in which they appear in the consensus mutation file produced by the MEMDS analysis pipeline.

2) **Mut\_per\_position\_all.txt** - A summary file containing substitution counts organized by substitution type and by read-reference alignment position. Contains the following columns:

a) **read\_position** - position of given base in the read-reference alignment. First position of the alignment is counted as ‘1’.

b) **coding\_sequence\_position** - position of given base relative to the TSS (translation start site) in the reference gene. TSS coordinates are given by the “**coding\_start**” field in the **“summarize\_params.txt”** parameter table.

c) **sequence** - base in the reference sequence found at given read-reference alignment position. Reference sequence is taken from MEMDS analysis pipeline parameter files.

d) **count columns** -columns showing number of read families containing specific substitution type at given alignment position. The number of shown substitution types depends on the nucleotide alphabet listed in the “**bases**” field of the **“summarize\_params.txt”** parameter table. Default columns are “A”, “C”, “G” and “T”.

An example row from the file looks like this:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| read\_position | coding\_sequence\_position | sequence | A | C | G | T |
| 31 | 1 | A | 0 | 0 | 0 | 0 |

3) **Mut\_per\_position\_no\_exceptions.txt** - A summary file similar to the “**Mut\_per\_position\_all.txt”**, but excluding counts of mutation types defined in the “**subs\_excluded**” field of the **“summarize\_params.txt”** parameter table. Counts of these mutations are shown as ‘0’.

4) **Mut\_per\_position\_exceptions.txt** - A summary file similar to the “**Mut\_per\_position\_all.txt”**, but including only counts of mutation types defined in the “**subs\_excluded**” field of the **“summarize\_params.txt”** parameter table. Counts of all other mutations are shown as ‘0’.

The goal of the tables above is to make it easier for user to parse substitution counts observed in the data, by type or position, presenting them in a convenient tabular format.

**2) Create a table of counted substitutions per cut-off criteria set  
Step description:**This script uses ”**Mut\_per\_position\_all.txt**” table to find counts and frequencies of substitutions of different types in the analyzed data using user-defined parameters for positions and substitution types to be excluded from the analysis. This allows the users to investigate analyzed genes using different parameters than those defined in the **“summarize\_params.txt”** parameter table, without the need to re-run the ”**consensus-count\_summary.sh**” script each time the parameters are modified, which can be highly time-consuming. Additionally, the script summarizes in a single table substitution counts obtained under different cut-off criteria used to find consensus mutations, allowing the users to investigate the effects of cut-off criteria on data.

**Substitution counts** represent number of substitutions of given type found in the user specified part of the read-reference alignment, when accounting for excluded positions and substitution types. **Substitution frequencies** are calculated as:

**Whereby:**

*- substitution\_count* - count of substitutions of given type within user-defined portion of the alignment, while **excluding** specific positions and substitution types defined by the user.

*- base\_number* - number of base pairs within user-defined portion of the alignment that can potentially create given substitution type. E.g.: for ‘CA’ substitution it would be number of cytosine bases within user specified coordinates.

*- family\_number* ***-*** count of “good” read families (not ambiguous and plasmids), provided by the ”**consensus-count\_summary.sh**” script output.

Substitution types analyzed by the script are defined by the nucleotide alphabet listed in the **“bases”** field of the **“summarize\_params.txt”** file. The script generates all possible combinations of provided bases, and outputs substitution counts and frequencies for each of them, except those excluded by the user-defined parameters.

**Run instructions:**a) Check that the **“factors\_table.txt”** contains correct values in the columns: “sort\_refs” and “sort\_ref”. In case of incorrect values - refer to the “MEMDS\_pipeline\_guide” regarding parameter file preparation. Check that **“summarize\_params.txt”** parameter table contains correct data and is placed in the “config\_files” folder, as described above.

b) **Run script:** *bash substitution\_collate.sh 1 $genes\_analyzed $excl\_pos $excl\_sub $excl\_var*

**User-defined parameters:**

*$genes\_analyzed* ***-*** Names of the analyzed genes, should match the names listed in the first column of **“summarize\_params.txt”** table.Multiple genes can be supplied as a comma-separated list (e.g.: HBB,HBD,...).

*$excl\_pos* - Positions in the alignment to exclude when counting substitutions, comma-separated (e.g.: 39,52,...).

*$excl\_sub* - Substitution types to exclude from substitution counts, comma-separated (e.g.: CA,GA,...).

*$excl\_var* - Specific substitutions to exclude from substitution counts, comma-separated (e.g.: 39CT,47GA,...).

c) Wait for the job to end. Use the “sacct” and the “squeue” commands on the cluster to check job status.

d) Inspect the output in the “tables\_consensus.BC3cutoff$BC3\_min\_cutoff. substitutions” directory, whereby “BC3\_min\_cutoff” is defined by the “BC3\_min” parameter in the “params\_1.sh” file.

**Output description:**

This step outputs counts and frequencies of substitutions in the analyzed data, while excluding user-specified alignment positions and substitution types from the analysis. Under the main output directory a sub-folder is created for each set of user-defined parameters used to run the script. Sub-folder name lists the parameters for excluded positions, substitution types and specific substitutions (e.g.: excl\_pos\_NONE\_subs\_CA\_GA\_muts\_NONE). Upon its completion following files are added to the relevant output folder:

1) **<sample\_name>.<ref\_name>.bwa.sorted.bam.subs.err,**

**<sample\_name>.<ref\_name>.bwa.sorted.bam.subs.out** - Log files produced during the run of the job on the cluster. Added to the “logs” folder inside output directory. Check that the “.err” file doesn’t contain error messages and that the “.out” file contains correct data for run parameters and indicates that the job was completed successfully.

2) **<sample\_name>\_<ref\_name>\_all.txt, <sample\_name>\_<ref\_name>\_NRS.txt, <sample\_name>\_<ref\_name>\_RS.txt** - Summary tables of substitution counts and frequencies. “All” tables include counts and frequencies across entire analyzed region of the alignment. “NRS” tablesinclude counts and frequencies inside ROI-flanking regions. “RS” tablesinclude counts and frequencies inside ROI. The tables contain the following columns:

a) **Name** - name of the input data table used in the analysis in a format of <sample\_name>\_<ref\_name> (e.g.: Apl\_cont\_idx0\_APL1).

b) **MutFreq** - mutation frequency threshold.

c) **ReadCount** - read family size threshold.

d) **BC3\_groups** - threshold of different 3’ barcode groups associated with the allele within the read family.

e) **BC3\_above** - threshold of different 3’ barcode groups associated with the allele within the read family that have above user-specified number of reads in each group.

f) **Family\_counts** - count of “good” read families that pass the combined cut-off criteria listed in the columns above and are used for finding variant alleles.

g) **Substitution counts** - the columns following “Family\_counts” column list counts of substitutions found in the data when the cut-off criteria listed above are applied. Each column lists counts for different substitution type, with substitution name listed in the column header.

h) **Substitution frequencies** - the columns following substitution count columns list frequencies of substitutions found in the data when the cut-off criteria listed above are applied. Each column lists frequencies for a different substitution type, with substitution name listed in the column header.

**3) Create a table of expected and counted substitutions inside ROI per cut-off criteria set  
Step description:**This script uses ”**Mut\_per\_position\_all.txt**” table to find expected and observed substitution counts inside ROI (region of interest) per cut-off criteria set. Under the assumption that mutations found in the ROI-flanking regions of the alignment are likely to represent artifacts, their count can be used to predict how many substitutions of a given type found inside ROI might be some sort of sequencing noise rather than true variants.

The expected artifact substitutions inside the ROI can be quantified as follows:

**Whereby:**

*- substitution\_count\_flanks* - count of substitutions of given type within ROI-flanking portion of the alignment, while **excluding** specific positions and substitution types defined by the user.

*- base\_number\_flanks* - number of base pairs within ROI-flanking regions of the alignment that can give rise to a given substitution type. E.g.: for ‘CA’ substitution it would be number of cytosine bases within user specified coordinates.

*- base\_number\_ROI* - number of base pairs within ROI that can give rise to a given substitution type.

If expected substitution counts inside ROI for given substitution type are similar to the observed ones, there is a high probability that observed substitutions inside ROI might be sequencing artifacts. If observed counts are higher than expected ones, this might point towards them being true variants. If the difference is particularly high, this can indicate that analyzed gene had variant base at this position to begin with and the variant does not represent *de novo* mutation. If expected counts are visibly higher than observed, this might indicate that ROI-flanking regions are particularly “noisy” in terms of given substitution type. In such a case it is worth investigating if few specific bases are main contributors to this “noise” and consider their exclusion from various substitution count analyses.

As the substitution count script above, this script utilizes user-defined parameters for positions and substitution types to be excluded from the analysis to allow higher analysis flexibility. The script summarizes in a single table data obtained under different cut-off criteria used to find consensus mutations, allowing the users to investigate the effects of cut-off criteria on data. Substitution types analyzed by the script are defined by the nucleotide alphabet listed in the **“bases”** field of the **“summarize\_params.txt”** file. The script generates all possible combinations of provided bases and outputs expected and observed substitution counts for each of them, except those excluded by the user-defined parameters.

**Run instructions:**a) Check that the **“factors\_table.txt”** contains correct values in the columns: “sort\_refs” and “sort\_ref”. In case of incorrect values - refer to the “MEMDS\_pipeline\_guide” regarding parameter file preparation. Check that **“summarize\_params.txt”** parameter table contains correct data and is placed in the “config\_files” folder, as described above.

b) **Run script:** *bash expected\_mut.sh 1 $genes\_analyzed $excl\_pos $excl\_sub $excl\_var*

**User-defined parameters:**

*$genes\_analyzed* ***-*** Names of the analyzed genes, should match the names listed in the first column of **“summarize\_params.txt”** table.Multiple genes can be supplied as a comma-separated list (e.g.: HBB,HBD,...).

*$excl\_pos* - Positions in the alignment to exclude when counting substitutions, comma-separated (e.g.: 39,52,...).

*$excl\_sub* - Substitution types to exclude from substitution counts, comma-separated (e.g.: CA,GA,...).

*$excl\_var* - Specific substitutions to exclude from substitution counts, comma-separated (e.g.: 39CT,47GA,...).

c) Wait for the job to end. Use the “sacct” and the “squeue” commands on the cluster to check job status.

d) Inspect the output in the “tables\_consensus.BC3cutoff$BC3\_min\_cutoff.exp\_sub” directory, whereby “BC3\_min\_cutoff” is defined by the “BC3\_min” parameter in the “params\_1.sh” file.

**Output description:**

This step outputs counts expected and observed substitutions inside ROI, grouped by substitution type. Under the main output directory a sub-folder is created for each set of user-defined parameters used to run the script. Sub-folder name lists the parameters for excluded positions, substitution types and specific substitutions (e.g.: excl\_pos\_NONE\_subs\_CA\_GA\_muts\_NONE). Upon its completion following files are added to the relevant output folder:

1) **<sample\_name>.<ref\_name>.bwa.sorted.bam.muts.err,**

**<sample\_name>.<ref\_name>.bwa.sorted.bam.muts.out** - Log files produced during the run of the job on the cluster. Added to the “logs” folder inside output directory. Check that the “.err” file doesn’t contain error messages and that the “.out” file contains correct data for run parameters and indicates that the job was completed successfully.

2) **<sample\_name>\_<ref\_name>\_expected\_subs.txt** - Summary tables of expected and observed substitution counts inside ROI. The tables contain the following columns:

a) **Name** - name of the input data table used in the analysis in a format of <sample\_name>\_<ref\_name> (e.g.: Apl\_cont\_idx0\_APL1).

b) **MutFreq** - mutation frequency threshold.

c) **ReadCount** - read family size threshold.

d) **BC3\_groups** - threshold of different 3’ barcode groups associated with the allele within the read family.

e) **BC3\_above** - threshold of different 3’ barcode groups associated with the allele within the read family that have above user-specified number of reads in each group.

f) **Family\_counts** - count of “good” read families that pass the combined cut-off criteria listed in the columns above and are used for substitution counts.

g) **Substitution counts** - the columns following “Family\_counts” column list expected and observed substitution counts. Each column lists counts for a different substitution type, with substitution name listed in the column header. The expected counts have “e” suffix in the column header (from “**e**xpected”), e.g.: “CA\_e”. The observed counts have “c” suffix in the column header (from “**c**ounted”), e.g.: “CA\_c”. The columns have interleaved order: first - column with expected counts for a given substitution type, than with observed counts, then with expected counts for a second substitution type and so on.

**4) Create a table of expected error and mutation rates per cut-off criteria set  
Step description:**This script summarizes error and mutation rates calculated by the **“consensus-count\_summary\_v3.R”** using different cut-off criteria into a single table, with each row representing values obtained under different cut-offs. This allows the user to check how cut-off criteria affect the data and choose which cut-offs should be used for further analyses of observed mutations.

**Run instructions:**a) Check that the **“factors\_table.txt”** contains correct values in the columns: “sort\_refs” and “sort\_ref”. In case of incorrect values - refer to the “MEMDS\_pipeline\_guide” regarding parameter file preparation.

b) **Run script:** *bash err\_rate\_collate.sh 1*

c) Wait for the job to end. Use the “sacct” and the “squeue” commands on the cluster to check job status.

d) Inspect the output in the “tables\_consensus.BC3cutoff$BC3\_min\_cutoff.err\_rate” directory, whereby “BC3\_min\_cutoff” is defined by the “BC3\_min” parameter in the “params\_1.sh” file.

**Output description:**

This step outputs summary of calculated error and mutation rates. Upon its completion following files are added to the output directory:

1) **<sample\_name>.<ref\_name>.bwa.sorted.bam.err.err,**

**<sample\_name>.<ref\_name>.bwa.sorted.bam.err.out** - Log files produced during the run of the job on the cluster. Added to the “logs” folder inside output directory. Check that the “.err” file doesn’t contain error messages and that the “.out” file contains correct data for run parameters and indicates that the job was completed successfully.

2) **<sample\_name>\_<ref\_name>\_error.txt** - Summary tables of error rates. The tables contain the following columns:

a) **Name** - name of the input data table used in the analysis in a format of <sample\_name>\_<ref\_name> (e.g.: Apl\_cont\_idx0\_APL1).

b) **MutFreq** - mutation frequency threshold.

c) **ReadCount** - read family size threshold.

d) **BC3\_groups** - threshold of different 3’ barcode groups associated with the allele within the read family.

e) **BC3\_above** - threshold of different 3’ barcode groups associated with the allele within the read family that have above user-specified number of reads in each group.

f) **Family\_counts** - count of “good” read families that pass the combined cut-off criteria listed in the columns above and are used for error rate calculations.

g) **WT** - error rate values obtained using cut-off criteria listed above, when calculation is done based on read family count of WT families only.

h) **WT+subs** - error rate values obtained using cut-off criteria listed above, when calculation is done based on read family count of WT and substitution containing families.

i) **WT+subs+indel** - error rate values obtained using cut-off criteria listed above, when calculation is done based on read family count of WT, substitution- and indel-containing families.

Overall, the error rate tends to become smaller when more stringent cut-off criteria are applied, as can be expected. However, increased stringency also means that more read families are removed from the analysis, since they do not pass the cut-off criteria. To test how much data is retained under different cut-off criteria, **“Family survival”** calculation was used. We took the data from “**Family\_counts**”column and for each cut-off criteria set we checked how many read families pass it, relative to the maximal number of families that pass any of the cut-off sets (max “**Family\_counts**” value). This analysis is not included with the script, on the account that the users might want to use their own calculations to estimate potential data loss with increased cut-off stringency.

3) **<sample\_name>\_<ref\_name>\_muts.txt** - Summary tables of corrected mutation rates. These tables have the same format as the error rate summary tables described above.  
Since substitutions inside ROI are expected to represent true variants, they are more likely to pass more stringent cut-off criteria than sequencing artifacts that often appear at low frequencies inside read families. But using too stringent cut-offs might start removing good variants as well. By looking at corrected mutation rates obtained for different cut-off criteria sets, the user can assess which cut-offs provide the optimal balance between noise removal and retention of true variants.

**5) Recalculate expected error and mutation rates using custom parameters  
Step description:**This script uses ”**Mut\_per\_position\_all.txt**” table to recalculate error and mutation rates per cut-off criteria set using user-defined parameters for alignment positions and substitution types that should be excluded from calculation. This allows easy adjustment of the calculations without the need to re-run the complete analysis done by the **“consensus-count\_summary\_v3.R”** script.

**Run instructions:**a) Check that the **“factors\_table.txt”** contains correct values in the columns: “sort\_refs” and “sort\_ref”. In case of incorrect values - refer to the “MEMDS\_pipeline\_guide” regarding parameter file preparation.

b) **Run script:** *bash re\_calc\_error\_rate.sh 1 $genes\_analyzed $excl\_pos $excl\_sub $excl\_var*

**User-defined parameters:**

*$genes\_analyzed* ***-*** Names of the analyzed genes, should match the names listed in the first column of **“summarize\_params.txt”** table.Multiple genes can be supplied as a comma-separated list (e.g.: HBB,HBD,...).

*$excl\_pos* - Positions in the alignment to exclude when counting substitutions, comma-separated (e.g.: 39,52,...).

*$excl\_sub* - Substitution types to exclude from substitution counts, comma-separated (e.g.: CA,GA,...).

*$excl\_var* - Specific substitutions to exclude from substitution counts, comma-separated (e.g.: 39CT,47GA,...).

c) Wait for the job to end. Use the “sacct” and the “squeue” commands on the cluster to check job status.

d) Inspect the output in the “tables\_consensus.BC3cutoff$BC3\_min\_cutoff. re-calc\_err” directory, whereby “BC3\_min\_cutoff” is defined by the “BC3\_min” parameter in the “params\_1.sh” file.

Under the main directory separate folder is generated for each combination of user-defined parameters used with the script, having a format: “**excl\_pos\_X\_subs\_Y\_muts\_Z**”. The output tables are stored under these sub-folders.

**Output description:**

This step outputs error and mutation rate tables, recalculated with user-defined parameters. Upon its completion following files are added to the output directory:

1) **<sample\_name>.<ref\_name>.bwa.sorted.bam.re-err.err,**

**<sample\_name>.<ref\_name>.bwa.sorted.bam.re-err.out** - Log files produced during the run of the job on the cluster. Added to the “logs” folder inside output directory. Check that the “.err” file doesn’t contain error messages and that the “.out” file contains correct data for run parameters and indicates that the job was completed successfully.

2) **<sample\_name>\_<ref\_name>\_re-err.txt,**

**<sample\_name>\_<ref\_name>\_re-mut.txt** - Summary tables of re-calculated error and mutation rates. The tables have the same format as the summary tables of error and mutation rates generated by the “**err\_rate\_collate.R**” script above.

**6) Plot substitutions per alignment position, as a percent of all substitutions  
Step description:**This script uses ”**Mut\_per\_position\_all.txt**” table to plot substitution abundance at each position of the alignment for each analyzed set of cut-off criteria. Substitution abundance at alignment position is calculated as a percent out of total number of substitutions obtained for a given cut-off criteria set. Different substitutions occurring at same position are noted by different colors, listed in the chart legend. The plots represent a convenient way to compare substitution abundance inside and outside ROI and to notice unexpectedly “noisy” position in the ROI-flanking regions.  
The script utilizes user-defined parameters for positions and substitution types to be excluded from the analysis to allow higher analysis flexibility. If certain substitutions are found at particularly high abundance, they can mask distribution of less abundant variants. In such case it can be useful to generate the plots twice - with and without the high abundance substitutions, to allow a more detailed view of different variants.

**Run instructions:**a) Check that the **“factors\_table.txt”** contains correct values in the columns: “sort\_refs” and “sort\_ref”. In case of incorrect values - refer to the “MEMDS\_pipeline\_guide” regarding parameter file preparation.

b) **Run script:** *bash plot\_subs.sh 1 $genes\_analyzed $excl\_pos $excl\_sub $excl\_var*

**User-defined parameters:**

*$genes\_analyzed* ***-*** Names of the analyzed genes, should match the names listed in the first column of **“summarize\_params.txt”** table.Multiple genes can be supplied as a comma-separated list (e.g.: HBB,HBD,...)

*$excl\_pos* - Positions in the alignment to exclude when counting substitutions, comma-separated (e.g.: 39,52,...).

*$excl\_sub* - Substitution types to exclude from substitution counts, comma-separated (e.g.: CA,GA,...).

*$excl\_var* - Specific substitutions to exclude from substitution counts, comma-separated (e.g.: 39CT,47GA,...).

c) Wait for the job to end. Use the “sacct” and the “squeue” commands on the cluster to check job status.

d) Inspect the output in the “tables\_consensus.BC3cutoff$BC3\_min\_cutoff.plots” directory, whereby “BC3\_min\_cutoff” is defined by the “BC3\_min” parameter in the “params\_1.sh” file.

Under the main directory, separate folder is generated for each set of user-defined parameters used, having a format: “**excl\_pos\_X\_subs\_Y\_muts\_Z**”. Under it sub-folders are generated to store the output for each set of combined cut-off criteria, e.g.: “**mutFreq0.0\_readCount0\_BC3WithMut0\_BC3above0**”. Sub-folder name includes the following cut-off values:

a) **mutFreq<W>** - mutation frequency threshold.

b) **readCount<X>** - read family size threshold.

c) **BC3WithMut<Y>** - threshold of different 3’ barcode groups associated with the allele within the read family.

d) **BC3above<Z>** - threshold of different 3’ barcode groups associated with the allele within the read family that have above user-specified number of reads in each group.

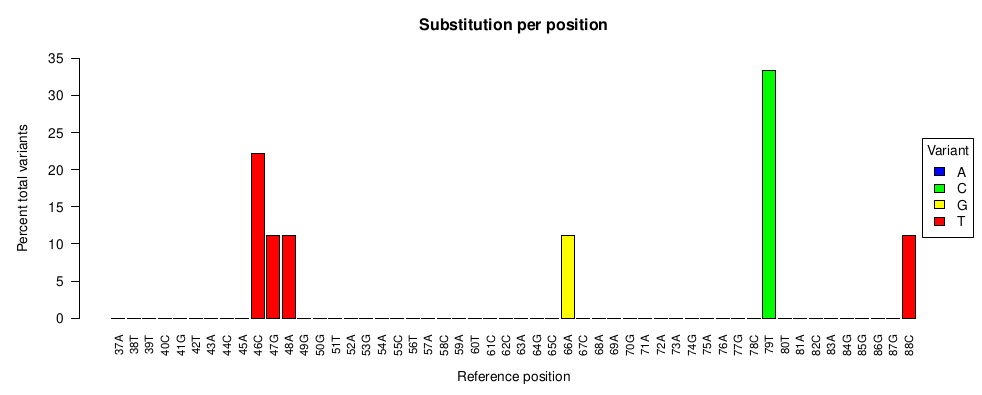
**Output description:**

This step outputs bar plots of substitution abundance per alignment position for each set of cut-off criteria used. Upon its completion following files are added to the output directory:

1) **<sample\_name>.<ref\_name>.bwa.sorted.bam.plots.err,**

**<sample\_name>.<ref\_name>.bwa.sorted.bam.plots.out** - Log files produced during the run of the job on the cluster. Added to the “logs” folder inside output directory. Check that the “.err” file doesn’t contain error messages and that the “.out” file contains correct data for run parameters and indicates that the job was completed successfully.   
The “.err” file sometimes can contain a warning that the script trying to create a folder that already exists. This is happening because outputs for control and experiment treatment data are stored in the same folder (for each cut-off criteria set). And if cluster jobs processing the control and the experiment data try to create the output folder concurrently - this warning is created. It has no bearing on the output and can be safely ignored.

2) **<sample\_name>\_<ref\_name>.variant\_plot.pdf** - Bar plots of mutation abundance per alignment position:



**7) Create a table of plasmid to WT ratios per cut-off criteria set  
Step description:**This script uses mutation count summary files (”**Mut\_totals\_summary.txt**”) created by the “**consensus-count\_summary\_v3.R**” script to calculate ratio of plasmid to WT read families for each analyzed set of cut-off criteria. The plasmid sequences, used in the MEMDS protocol to calculate enrichment factor, are designed to be processed during the experimental procedure similarly to the WT sequences of the analyzed genes. Therefore they are not expected to behave differently than WT read families under different cut-off criteria sets, and their ratio to WT families should remain stable. If the ratio starts to change after certain level of cut-off criteria stringency, that might indicate that given cut-off criteria are too demanding and less stringent criteria should be chosen for the analyses.

The script utilizes user-defined profiles of plasmid sequences to extract their counts from the summary files. Since the script is **extracting** information from existing summary files, and not making ***de novo*** counts, the profiles supplied to the script should represent one or more of the profiles appearing in the **“plasmid\_mut\_profile”** field of **“summarize\_params.txt”** parameter table.

**Run instructions:**a) Check that the **“factors\_table.txt”** contains correct values in the columns: “sort\_refs” and “sort\_ref”. In case of incorrect values - refer to the “MEMDS\_pipeline\_guide” regarding parameter file preparation.

b) **Run script:** *bash plasmid-WT\_ratio.sh 1 $genes\_analyzed $plasmid*

**User-defined parameters:**

*$genes\_analyzed* ***-*** Names of the analyzed genes, should match the names listed in the first column of **“summarize\_params.txt”** table.Multiple genes can be supplied as a comma-separated list (e.g.: HBB,HBD,...).

*$plasmid* - Plasmid sequence profiles distinguishing plasmids from the sequences of the analyzed gene(s). Should match one or more of the profiles appearing in the **“plasmid\_mut\_profile”** field of **“summarize\_params.txt”** parameter table. Multiple profiles should be separated by semi-colon, with individual positions of the same profile separated by commas, e.g.: **39CT,46C-,47C-;39CT,48T-,51-A**.

c) Wait for the job to end. Use the “sacct” and the “squeue” commands on the cluster to check job status.

d) Inspect the output in the “tables\_consensus.BC3cutoff$BC3\_min\_cutoff.

plasmid-WT\_ratio” directory, whereby “BC3\_min\_cutoff” is defined by the “BC3\_min” parameter in the “params\_1.sh” file.

**Output description:**

This step outputs ratios of plasmid to WT read families’ counts for each analyzed set of cut-off criteria. Upon its completion following files are added to the output directory:

1) **<sample\_name>.<ref\_name>.bwa.sorted.bam.ratio.err,**

**<sample\_name>.<ref\_name>.bwa.sorted.bam.ratio.out** - Log files produced during the run of the job on the cluster. Added to the “logs” folder inside output directory. Check that the “.err” file doesn’t contain error messages and that the “.out” file contains correct data for run parameters and indicates that the job was completed successfully.

2) **<sample\_name>\_<ref\_name>\_plasmid-WT\_ratio.txt** - Summary tables of of plasmid to WT read families’ counts ratio. The tables contain the following columns:

a) **Name** - name of the input data table used in the analysis in a format of <sample\_name>\_<ref\_name> (e.g.: Apl\_cont\_idx0\_APL1).

b) **MutFreq** - mutation frequency threshold.

c) **ReadCount** - read family size threshold.

d) **BC3\_groups** - threshold of different 3’ barcode groups associated with the allele within the read family.

e) **BC3\_above** - threshold of different 3’ barcode groups associated with the allele within the read family that have above user-specified number of reads in each group.

f) **WT** - count of WT read families that pass the combined cut-off criteria listed in the columns above and are used for ratio counts.

g) **Plasmid counts** - the columns following the ‘WT’ counts column show counts of detected plasmid families for each set of cut-off criteria listed in the columns above. The number of columns depends on the number of distinct plasmid profiles supplied by the user. For each plasmid profile two columns are created: first shows counts of plasmid families when mismatches are allowed between the plasmid profile and the mutations observed in the read and the second shows counts of exact matches only (see explanation on the output of “**consensus-count\_summary\_v3.R**” script above for more information on plasmid family counts). Column headers have a format of “P#”, where “#” is the ordinal number of given plasmid profile, according to the order in which the profiles are given in the *$plasmid* parameter of the script. Columns showing exact counts have the “exact\_” prefix in their header (e.g.: “P1” is followed by “exact\_P1”).

h) **Plasmid to WT ratios** - the columns following plasmid counts show the calculated ratios between the plasmid and the WT read families for each set of cut-off criteria listed in the columns above. Column headers have the name of plasmid count column on which the calculation is based and “-WT\_ratio” suffix (e.g.: “P1-WT\_ratio” or “exact\_P1-WT\_ratio”).

**Primary barcode nucleotide composition analysis**

**1) Check nucleotide frequencies within primary barcodes to assess barcode diversity  
Step description:**This scripttakes primary barcode sequences of read families used for mutation calling by the MEMDS analysis pipeline and checks frequency of “A”, “C”, “T” and “G” nucleotides in them. Read families are grouped by family size (number of reads in the family) and nucleotide frequencies are reported as mean and median values per family size group.

The accuracy of MEMDS analysis is dependant on proper barcoding of the analyzed DNA fragments. To this end the library of barcodes used in the experiment should be sufficiently randomized. In a theoretical fully randomized pool of barcode sequences frequencies of individual nucleotides within the sequences are expected to be about equal. However, methods used today to create randomized oligo sequences have various biases, causing some nucleotides to appear more frequently than others in the resulting sequences. Still, in a properly randomized barcode library it is rather uncommon to have large scale biases, like having single nucleotide at 50% or more of barcode positions. While a biased library of barcode sequences might still be diverse enough for analysis purposes, the results obtained with such a library should be carefully examined for presence of unexpected behaviors.

**Run instructions:**a) Check that the **“factors\_table.txt”** contains correct values in the columns: “size\_f”, “read\_seq”, “read\_pos”, “sort\_refs” and “sort\_ref”. In case of incorrect values - refer to the “MEMDS\_pipeline\_guide” regarding parameter file preparation.

b) **Run script:** *bash BC5\_nucleotide\_dist.sh 1 $genes\_analyzed $id\_len*

**User-defined parameters:**

*$genes\_analyzed* ***-*** Names of the analyzed genes, should match the names listed in the first column of **“summarize\_params.txt”** table.Multiple genes can be supplied as a comma-separated list (e.g.: HBB,HBD,...).

*$id\_len* - length of the sample identifier portion of the primary barcode which is constant between all barcodes of the same sample. This portion of barcode sequence is excluded from the analysis since it is not contributing to the diversity of barcode pool.

c) Wait for the job to end. Use the “sacct” and the “squeue” commands on the cluster to check job status.

d) Inspect the output in the “tables.BC3cutoff$BC3\_min\_cutoff.BC5\_nucl” directory, whereby “BC3\_min\_cutoff” is defined by the “BC3\_min” parameter in the “params\_1.sh” file.

**Output description:**

This step reports nucleotide frequencies in the primary barcodes. Upon its completion following files are added to the output directory:

1) **<sample\_name>.<ref\_name>.bwa.sorted.bam.BC5\_log.err,**

**<sample\_name>.<ref\_name>.bwa.sorted.bam.BC5\_log.out** - Log files produced during the run of the job on the cluster. Added to the “logs” folder inside output directory. Check that the “.err” file doesn’t contain error messages and that the “.out” file contains correct data for run parameters and indicates that the job was completed successfully.

2) **<sample\_name>.<ref\_name>.bwa.sorted.bam.BC5\_nucl.txt** - Tab-delimited summary table outlining analysis results. Contains the following columns:

a) **Family\_size** - number of reads in given read families.

b) **Family\_count** - number of read families of given size.

c) **Nucleotide frequency columns** - the columns following the **“Family\_count”** column list frequencies of nucleotides in the read families of given size, in the order of “A”, “C”, “T” and “G”. For each nucleotide three values, listed in separate columns, are calculated: mean frequency of the nucleotide, median frequency of the nucleotide and standard deviation of frequency values across primary barcodes belonging to read families of given size.

**Relabeling incidence analysis**

**1) Check frequency of relabeling across read families  
Step description:**This scriptassesses incidence of relabeling of the analyzed reads, by searching for presence of the relabeling oligo sequence among reads that were filtered by the MEMDS analysis pipeline for not having correct sample-identifier sequences. The relabeling oligo is identical in sequence to the secondary barcode primers except for the sample-identifier and the secondary barcode features that are replaced by a known sequence. Therefore, in the event of incomplete degradation by the 3’-exonuclease, the amount of reads with relabeling sequence signature serves as a proxy for the frequency of relabeling by the secondary barcode primer.

**Run instructions:**a) Check that the **“factors\_table.txt”** contains correct values in the columns: “size\_f”, “read\_seq”, “read\_pos”, “sort\_refs” and “sort\_ref”. In case of incorrect values - refer to the “MEMDS\_pipeline\_guide” regarding parameter file preparation.

b) **Run script:** *bash relabeling\_dist.sh 1 $genes\_analyzed $oligoD $id\_len*

**User-defined parameters:**

*$genes\_analyzed* ***-*** Names of the analyzed genes, should match the names listed in the first column of **“summarize\_params.txt”** table.Multiple genes can be supplied as a comma-separated list (e.g.: HBB,HBD,...).

*$oligoD* - Relabeling sequence signature.

*$id\_len* - length of the sample identifier portion of the primary barcode which is constant between all barcodes of the same sample. Used to identify reads that have a good primary barcode with correct sample ID, but malformed secondary barcode due to relabeling.

c) Wait for the job to end. Use the “sacct” and the “squeue” commands on the cluster to check job status.

d) Inspect the output in the “tables.BC3cutoff$BC3\_min\_cutoff.relabeling” directory, whereby “BC3\_min\_cutoff” is defined by the “BC3\_min” parameter in the “params\_1.sh” file.

**Output description:**

This step reports relabeling incidence among analyzed reads. Upon its completion following files are added to the output directory:

1) **<sample\_name>.<ref\_name>.bwa.sorted.bam.relabeling.err,**

**<sample\_name>.<ref\_name>.bwa.sorted.bam.relabeling.out** - Log files produced during the run of the job on the cluster. Added to the “logs” folder inside output directory. Check that the “.err” file doesn’t contain error messages and that the “.out” file contains correct data for run parameters and indicates that the job was completed successfully.

2) **<sample\_name>.<ref\_name>.bwa.sorted.bam.BC5\_nucl.txt** - Tab-delimited summary tables outlining analysis results. Summary files contain two tables, situated one below the other.

The first table shows distribution of relabeled reads and contains the following columns:

a) **Family\_size** - in analogy with “good” read families indicates size of relabeled read families - groups of reads sharing same primary barcode and having relabeling oligo at the place of their secondary barcode.

b) **Family\_count\_all** - count of relabeled read families of given size.

c) **Family\_count\_cleaned** - MEMDS analysis pipeline filters out and lists in a separate file barcodes of reads not having proper sample-identifier either in the primary or secondary barcodes. Therefore some of the reads having relabeling oligo instead of proper secondary barcode might also have malformed primary barcode. This makes it problematic to group such reads into families based on their primary barcode. To show that the distribution of relabeled read families’ sizes is not an artifact of malformed primary barcodes, the script provides also counts of relabeled read families having only good primary barcodes, with correct sample ID, for each family size category.

d) **Family\_perc\_all** - percent of relabeled read families of given size out of total number of relabeled read families.

e) **Family\_perc\_cleaned** - percent of relabeled read families of given size out of total number of relabeled read families, when only families with a good primary barcode are counted.

The second table shows relationship between relabeled and “good” read families (with correct sample identifiers in both primary and secondary barcodes) and contains the following columns:

a) **Family\_size\_mut** - size of “good” read families sharing primary barcode with one or more relabeled reads.

b) **Family\_count\_labeled** - count of “good” read families of given size that share primary barcode with relabeled reads.

c) **Family\_count\_all** - count of all “good” read families of given size in the analyzed data, whether they are related to relabeled reads or not.

d) **Family\_perc\_labeled** - percent of “good” read families of given size out of total number of “good” read families, when only read families sharing primary barcode with relabeled reads are considered (based on “**Family\_count\_labeled**” column).

e) **Group\_frequency** - ratio between “good” read families sharing primary barcode with relabeled reads and all “good” read families in each family size category (calculated as **Family\_count\_labeled/Family\_count\_all**).