**Introduction**

The MEMDS analysis pipeline is an accompanying software package to the MEMDS sequencing protocol. The pipeline analyses deep sequencing data produced by MEMDS, and outputs summary tables of mutations found in the analyzed data relative to the reference gene(s).

**Currently, the pipeline is CLI (command line interface) only**, thus its scripts need to be executed from the terminal of operational system on which it is ran.

**Quick start guide**  
This section provides basic information needed to run the pipeline. For a more detailed information regarding each analysis step and its associated output refer to the next section.

**Note:** Here the “**$i”** symbol is used to indicate that numerical option 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. To complete the step, **my\_script.sh** should be run with all the sub-steps in a sequential order, save for steps marked as “optional” in the guide below.

**Note-2:** When running on a cluster, for each job submitted by the script the following message would appear: “Submitted batch job #job\_serial\_number”.

**Before moving to the next option in the script or to the next step in the pipeline**, a**lways check that all running jobs were completed successfully.** To check job status, use the following commands (for SLURM systems):

1) *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.

2) *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!

**Note-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.

In addition, 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.

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 pipeline steps and choose which step to execute.

To run the wrapper script, navigate from the terminal to the directory containing the helper script and use the command: *bash MEMDS\_pipeline\_wrapper.sh.* By default, the wrapper script is located in the directory of the pipeline scripts, but can be placed and run from any place on the machine. **Note:** if pipeline 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 pipeline steps to run. If a step contains several sub-steps, additional menu would appear asking to choose relevant sub-step. If multiple paths are provided, step 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) **(Optional) Merging raw data:**   
 a) *bash concatenate\_partfiles.sh*   
 b) **Local run:** *bash fastq\_merging/samples\_table.concat.sh* **or**

c) **Cluster run:** *srun bash fastq\_merging/samples\_table.concat.sh*

3) **Formatting parameter data for use by the pipeline:**

a) **Single-end data:** *bash setting\_1-SE.sh* **or**

b) **Paired-end data:** *bash setting\_1-PE.sh*

4) **Quality control and clearing of raw data + paired-end data merging**:  
a) **Single-end data:** *bash filter-SE4.sh $i* (1-3) **or**

b) **Paired-end data:** *bash filter-PE4.sh $i* (1-4)

5) **Selecting reads with correct barcodes and separating between barcodes and genomic data**:

a) *bash trim7.sh 1*

6) **Sorting reads by their origin gene:**

a) *bash sort2.sh 1*

7) **Mapping reads to the reference sequences:**

a) *bash bwa9.sh $i* (1-2)

8) **Making alignment files viewable in IGV:**

a) *bash create\_dummy\_genome5.sh 1*

9) **Creating mutation table that lists sequencing quality alongside each mutation:**

a) *bash sam\_to\_mutation-list-3.sh 1*

10) **Creating a table of mutations found in the analyzed data:**

a) *bash sam\_to\_mutation-table\_5.2.sh 1*

11) **Detecting consensus mutations that pass a set of user defined thresholds:**

a) *bash consensus\_15.1.sh $i* (1-2)