GENOMICON-Seq V.1.0. : GENOmics Modeling, In silico CONstruction and Sequencing

Table of Content

I) Introduction

Welcome to GENOMICON-Seq, a tool for genetic exploration through simulation. GENOMICON-Seq is designed to simulate both amplicon sequencing and whole exome sequencing (WES), providing a robust platform for users to experiment with virtual genetic samples.

At the core of GENOMICON-Seq is the ability to generate samples with varying mutation frequencies, which are then subjected to a simulated library preparation process. This unique feature allows users to introduce and adjust the level of noise within each simulation, effectively masking the detection of real mutations to various extents. Such simulations are critical in demonstrating how different noise levels can obscure mutation detection, offering valuable insights for refining laboratory techniques and enhancing mutation analysis.

Users can explore a range of different scenarios facilitating a deeper understanding of the dynamic interplay between genetic variations and the technical limitations of sequencing technologies.

Dive into GENOMICON-Seq and harness the power of simulation to advance your mutation detection strategies and guide improvements in laboratory practices.

The GENOMICON-Seq allows users to:

a) control the generation of the initial sample with specified fasta sequences, their copy numbers, and mutation insertions. Users can define the mutation level, mutation targets, and substitution types thereby fully controlling the mutational process;

b) control the fraction of the initial sample that will be fragmented and the length of the produced fragments. As each fragmentation is a unique random process, users can specify the number of fragmentation replicates generated for each sample.

c) chose between amplicon or whole exome sequencing.

   1. In amplicon sequencing users control the number of simulated PCR reactions for each fragmented replicate, the polymerase error rate in the PCR reactions, the number of cycles and the amplification efficiency.
   2. In WES users control the matching length between the probe and simulated fragments that influences the selection of fragments for sequencing. In addition, a simplified PCR mimics the short indexing PCR excluding the polymerase error rate.

d) control the sequencing process, by specifying the number of reads that will be produced, and whether the fragment length will impact their probability of being sequenced (shorter fragment - higher probability of being sequenced). We modified the InSilicoSeq tool enabling it to produce the reads from generated fragments but preserving its ability to simulate the sequencing error of different Illumina instruments.

The simulation is based on many random processes that modulate the final results according to user specifications, but it is still 100% reproducible.

The manual will introduce users to the quick installation and setup, the parameters and their usage for both amplicon sequencing and WES, main input, and output files. This will be followed by the theory and processes behind the simulation, wrapping it up with some recommendations and warnings.

II) Setup and quick start

a. Setup

The GENOMICON-Seq represents a collection of scripts written in R, C++, PowerShell Python, and well-known bioinformatic tools, while the whole process is orchestrated by Snakemake. Currently, it is only available as a docker container. For a quick start, the command under will download and run the `genomiconseq_intall.sh` script that will, in turn, download the docker container, play dataset, Snakefiles and their config.yml files, and set everything up.

```
curl -L https://raw.githubusercontent.com/Rounge-lab/GENOMICON-
Seq/main/genomicon_setup.sh -o genomiconseq_install.sh
chmod +x genomicon_setup.sh
./genomicon_setup.sh
```

Use arguments --ampliseq or --wes (e.g . `./genomicon_setup.sh` --ampliseq when running the genomicon_setup.sh to download files/scripts necessary for either amplicon sequencing or WES simulation. Without the arguments, the script will download files/scripts for both simulations. The folder structure would be like this:

```
GENOMICON-Seq
├── config_ampliseq.yml
├── config_wes.yml
├── input_data_ampliseq
│   ├── all_primers.csv
│   ├── HPV16.fa
│   ├── HPV16.fa.fai
│   ├── multifasta_copy_number.csv
│   ├── primer_set_1.csv
│   ├── primer_set_2.csv
│   └── probability_variant_table.csv
├── input_data_wes
│   ├── chr1.fasta.gz
│   ├── dummy_probes.fasta
│   ├── fasta_lengths.csv
│   ├── multifasta_copy_number.csv
│   ├── probability_variant_table.csv
│   ├── SBS2_profile_short.csv
│   ├── xgen-exome-research-panel-v2-probes-hg38_filtered_chr1.bed
│   ├── xgen-exome-research-panel-v2-probes-hg38_short_all.bed
│   ├── xgen-exome-research-panel-v2-targets-hg38_filtered_chr1.bed
│       └── xgen-exome-research-panel-v2-targets-hg38_short_all.bed
├── Snakefile_ampliseq
├── Snakefile_wes
└── SQL_database
    ├── chr1.sqlite
    └── HPV16REF.sqlite
```

You can also download the docker container manually:

```
docker pull mimsto86/genomicon-seq:v1.0
```

The scripts necessary for running are integrated into the container, and they are available at https://github.com/Rounge-lab/GENOMICON-Seq/scripts

During the simulation run, SQL-databases for each sequence in a fasta file will be generated. While for small genomes (several Mb) the generation time will not be long, for WES simulation on a whole human genome, generation time might take several hours. The premade SQL-database for each hg38 chromosome is available for downloading here https://zenodo.org/records/12683302/files/SQL_database_human.tar.gz. The packed database size is 21.8 GB, when unpacked its size is 99.8 GB. Once made (or downloaded), the generation step of the SQL_database will be omitted as long as the headers of of the fasta sequences correspond to the names of the SQL-database.

We also recommend downloading the human genome (hg38) fasta file where chromosome headers correspond to the SQL-database names here: https://zenodo.org/records/12683302/files/human_genome.fasta.gz.

b. Quick start

As the simulation would require some substantial computational power (we will cover that later), a playset will be used to test if everything is functional and as an example throughout this manual. The genome for amplicon sequencing simulation is the human papillomavirus (HPV) type 16 genome, 7906bp. The parameters in the `confing_ampliseq.yml` are specified for testing purposes, each parameter will be explained in the next chapter. To run the simulation, run the following command from the main folder where the Snakemake_ampliseq and `config_ampliseq.yml` are:

```
docker run -it -v $(pwd):/usr/src/app/pipeline -w
/usr/src/app/pipeline mimsto86/genomicon-seq:v1.0 snakemake -j 1 -p
-s ./Snakefile_ampliseq --configfile ./config_ampliseq.yml
```

For simplicity and testing purposes, only human chromosome 1 will be used in the WES simulation. For the whole human genome, much greater computational power will be needed. The command for the test run is similar to the amplicon sequencing simulation.

```
docker run -it -v $(pwd):/usr/src/app/pipeline -w
/usr/src/app/pipeline mimsto86/genomicon-seq:v1.0 snakemake -j 1 -p
-s ./Snakefile_wes --configfile ./config_wes.yml
```

The main outputs are R1 and R2 fastq.gz files that can be found in Output_data_ampliconseq/sample_1/generated_reads or Output_data_wes/sample_1/generated_reads. We will have a separate chapter about the outputs and how they may serve you.

Each new simulation run must have its own output directory to avoid rewriting the current files and possible mixing-up of the generated outputs.

III) Simulation parameters

The **`config_ampliseq.yml`** and **`config_wes.yml`** are the main configuration files where parameters for both simulations can be specified. The parameters can be `optional` or `mandatory`, some of them are `mutually exclusive` and `advanced`, in the following text the same colour code would be used for different options. "Notes" would give some additional specifications for some parameters. In addition, certain parameters might be either amplicon sequencing or WES-specific (in the further text, files and outputs specific for amplicon sequencing or WES would have the same colour code). If not specified, the parameter is used in both simulations, not specified by colour.

General, sample, fragmentation, and sequencing parameters are mostly shared between amplicon sequencing and WES with some exceptions. Amplicon sequencing includes parameters for PCR simulation, while WES includes parameters for probe-capture enrichment and simplified PCR.

a. General parameters

1. `NUM_SAMPLES: {integer}` - specify the number of in-silico samples that will be produced. The parameter is `mandatory`.

2. `FRAG_REPLICATES: {integer}` - specify the number of fragmentation replicates. As fragmentation is a random process, the produced fragments will be different for each replicate. The parameter is `mandatory`.

3. `SEED: "{integer}"` - seed ensures that each simulation is reproducible. The parameter is `optional`. When not specified, the random seed will be assigned and recorded for each simulation.

4. `OUTPUT_DIRECTORY: "{absolute path/to/the/output}"` - specify the absolute path to the output directory with all the outputs. As the simulation was meant to be run from the docker container, please keep the current path `/usr/src/app/pipeline/Output_data`, the name of the Output_data can be edited. The parameter is `mandatory`.

Note 1: Each simulation run should have its unique output folder to avoid mixing the files and rewriting of the outputs, or the complete unwanted output folder should be removed before a new run with the same output directory!

5. `INPUT_DIRECTORY: “{name of the input folder}”` - The data playset is placed in the input_data folder specified under this parameter. Input data for amplicon sequencing is in the input_data_ampliseq while input data for WES is in input_data_wes. When using your data, feel free to edit the name of the INPUT_DIRECTORY but specify the directory name in the config file. The parameter is `mandatory`.
   Note 1: This parameter does not require the absolute path BUT only the name of the folder!

6. `FASTA_FILE: “{name of the fasta file}”` - specify the name of the fasta file (fa, fasta, fa.gz, fasta.gz formats are accepted). Multifasta files are also allowed. The parameter is `mandatory`.
   Note 1: The specified fasta file needs to be in the specified INPUT_DIRECTORY, the input files are described in the next chapter.

7. `CIRCULAR: “{Y|N}”` - specify the genome topology and can only be used in the amplicon sequencing config file. Genome can be specified as circular (Y) or linear (N). The parameter is `optional`, by default the genome is assumed to be circular.

b. Sample parameters

This parameter set modulates the composition of in-silico samples such as the number of genome copies, mutational target, substitution type and mutation distribution mode.

i. Number of genome copies in the sample

1. NR_COPIES: “{integer}” - specify the copy number of the genome(s) in the provided fasta file. If a multifasta file is specified, the NR_COPIES applies to each fasta sequence.
2. MULTIFASTA_COPIES_TABLE: “{table name}“ - specifies the table name containing the copy number of each genome/chromosome in the multifasta file. The format of the table will be described in the next chapter, Input Files.
   Note 1: Both parameters are `optional`. If neither NR_COPIES nor MULTIFASTA_COPIES is specified, the default number of copies is 1000 in Amplicon Sequencing and 100 in WES.

ii. Mutation targets

1. `MUT_CONTEXT: “{comma-separated string}”` - Specify nucleotide contexts (e.g., single nucleotide, trinucleotide or pentanucleotide context) that will be searched and mutated. In the case of trinucleotide or pentanucleotide sequences, the middle nucleotide will be mutated. Multiple nucleotides or

tri/penta contexts can specified, however, mixing them is not allowed. The parameter is `optional`, if it is not provided, any nucleotide in the sequence can mutate.
Note 1: Either multiple single nucleotides such as “A,T”, or multiple trinucleotide contexts “AGT,ACT”, or multiple pentanucleotide contexts “ATCGT,CGTAA” can be specified, but not a mix of them “ATC,ATGCT,A,ATC”.
Note 2: The specified context is searched on both lead and lag strands of fasta sequence(s).

iii. Substitution type

1. TS_TV_RATIO: “{integer:integer}” - specifies the transition vs transversion ratio (for example “3:1”). The specified ratio is converted into the probability of a nucleotide mutating to any other nucleotide.
2. `SUBSTITUTION_PROBABILITY_TABLE: “{csv table name}”` - the table allows complete control of the mutating processes, specifying the probability of substituting each nucleotide with another.
Note 1: `TS_TV_RATIO` and `VARIANT_PROBABILITY_TABLE` parameters are `optional` but `mutually exclusive`. When none of them is, by default, a nucleotide can mutate to any other nucleotide with equal probability.

iv. Mutation modes

Three mutational modes govern the mutation distribution on the genome copies, a) Deterministic mode, b) Specific mutation rate mode, and c) SBS mimicry. Deterministic and Specific mutation rate modes are mutually exclusive and their parameters should not be specified together. SBS-mimicry mode is an extension of the Deterministic mode requiring specification of its parameters and specified SBS signature table (COSMIC) used only in WES. In addition, all generated mutations will be found only in the exonic portion of the human genome. Chapter Theory and Processes behind the Simulation gives a detailed explanation of the mutation modes.

1. Deterministic mode

1. `MUT_GENOME_FRACTION: “{0 ≤ float ≤ 1}”` - specifies the fraction of all genome copies that will be mutated. If multifasta is used, this parameter is applied to all of them equally. This parameter is employed together with the `FRACTION_POSITIONS` parameter.

2. `FRACTION_POSITION: “{0 ≤ float ≤ 1}”`- specifies the fraction of all targeted positions that will mutate. If `MUT_CONTEXT` is not specified, the fraction of all positions will be used. The calculated number represents the maximum number of mutations specified fraction of genome copies (determined by the `MUT_GENOME_FRACTION`) will get by random chance. The parameter is `optional` and by default is 0.05 in amplicon seq and 0.005 in WES simulation.

Note 1: If MUT_CONTEXT is specified, only the positions matching the context will be taken into account when applying the FRACTION_POSTION. For example, if

ATC is targeted, with a total count of 100 in a specified genome (FASTA), and FRACTION_POSITION is set to 0.1, 10 out of 100 ATC positions will be selected for mutation. Each genome copy that will mutate (defined by MUT_GENOME_FRACTION) will get a randomly picked number of positions out of the 10 selected ATC positions.

2. *Specific mutation rate mode*

1. `MUTATION_RATE: “{scientific number, e.g. 1e-6}”` - specifies the mutation rate that together with the genome length and the number of genome copies in the sample determines how many positions on each genome copy will be mutated.
   Note 1: `MUT_GENOME_FRACTION + FRACTION_POSTION` and `MUTATION_RATE` are `mutually exclusive` and `mandatory`, one or the other must be specified!
   Note 2: In WES simulation, mutations will be exclusively generated in exons specified under the PROBE_TARGETS parameter (described under)!

3. *SBS-mimicry mode - available only in WES simulation*

1. `SBS_SIGNATURES: “{name of the table with SBS percentages}”` - This mode is the extension of the deterministic mode where `MUT_GENOME_FRACTION + FRACTION_POSTION` needs to be specified as well as the SBS table. The fraction of all exon positions will be mutated in a way that mimics the distribution given in the SBS table. The `FRACTION_POSITIONS` specifies the total number of mutations while the number of genomes the mutations will be randomly distributed across is given by `MUT_GENOME_FRACTION`. The `MUT_CONTEXT`, `TS_TV_RATIO`, and `VARIANT_PROBABILITY_TABLE` should also not be specified when the SBS table is given. For more detailed information read the chapters Input Files and Theory and Processes behind the Simulation.

c. Fragmentation parameters

Allows specification of a sample portion that will be used in the downstream simulated steps, and the length of the fragments to be produced. Note that if more than 1 fragmentation technical replicate is specified, these parameters will be applied to each replicate.

1. `FRAGMENTATION_FRACTION: “{0 ≤ float ≤ 1}”` specifies the fraction of all genome copies in a sample that will be fragmented. The parameter is `optional`, the default is 1 (100%).

2. `FRAGMENT_LENGTH: “{integer|integer:integer}”` - Available only in AmpliSeq - specifies the length of the fragments to be produced, it can be a single integer value where all produced fragments will have the same length, or it can be the range of values written as colon-separated integer values (e.g.

100:300). The parameter is `optional`, by default the fragment range will be 250:450.

3. `MIN_FRAGMENT_LENGTH: “integer”` - Available only in WES - specifies the minimum fragment length. The parameter is `mandatory`.

4. `MAX_FRAGMENT_LENGTH: “integer”` - Available only in WES - specifies the maximum fragment length. The parameter is `mandatory`.

d. Amplicon sequencing simulation - PCR parameters

The parameters specified here will modulate the PCR simulation. To get complete control over it and an understanding of the process, check the Chapter Theory and Processes behind the Simulation.

1. `PRIMERS: “{name of the csv table}”` - specifies the name of the csv table containing all the primers that will be used in the simulation, the format of the file is covered in the Input Files chapter! The parameter is `mandatory`.

2. `NUM_PCR: {integer}` - specifies the number of PCR reactions to be simulated. The parameter is `optional`, the default is 2.
   Note 1: For each specified PCR reaction, a csv file primer_set_*.csv must be found in the input directory, format and details about the csv file are covered in the Input Files section. These files don’t need to be specified anywhere, just need to exist in the input folder!

3. `AMPLICON_LENGTH: {integer}`- specifies the minimum amplicon length that will be produced, larger amplicons will not be produced. The parameter is `mandatory`.

4. `NUM_CYCLES: “{integer}”` - specifies the number of PCR cycles to be performed. The parameter is `optional`, the default number of PCR cycles is 25.

5. `POL_ERROR_RATE: “{scientific number, e.g. 1e-7}”` specifies the polymerase error rate. In each cycle, the number of new error mutations is calculated based on the current number of fragment copies and their lengths. A more detailed explanation of PCR simulation can be found in Theory and Processes behind the Simulation. The parameter is `optional`, the default polymerase error rate is set to 10^{-6} (1e-6).
   Note 1: Polymerase error rate can be set to 0.

6. `FRACTION_PCR_SEQUENCING: “{0 ≤ float ≤ 1}”` specifies the fraction of PCR reaction that will be sequenced. The parameter is `mandatory`.
   Note 1: Carefully calculate how much of your PCR reaction is sequenced and adjust the parameter value.

ADVANCED PARAMETERS

7. `K_PARAMETER_pcr: “{non-zero float}”` - k-parameter and midpoint cycle (next parameter) are responsible for the amplification efficiency drop during the PCR cycling modelled by the inverted sigmoid function. The k-parameter shapes the steepness of the drop. For more details check the chapter Theory and Processes behind the Simulation. The parameter is `optional`, the default value is 1.

8. `MIDPOINT_CYCLE: “{integer}”` - This parameter represents the PCR cycle at which the amplification efficiency is 50%. The parameter is `optional`, the default is 60% of the total number of specified PCR cycles.

e. WES - Probe-capture enrichment

In this section, probes and their targeted region will be specified. For more details about the formats of the files, read the chapter Input Files.

1. `PROBES_FASTA: “{name of the fasta file containing probe sequences}”` - Probes can be given in a FASTA format. The file should be placed in the `input_data_wes`.

2. `PROBES_BED: “{name of the BED file containing probe binding regions}”` - Probes can be given in BED format where columns specify the chromosome, start and end of the probe binding site. The file should be placed in `input_data_wes`. For testing purposes, xGen Exome Research Panel v2 probes are already placed in the input directory.
   Note1: PROBES_FASTA and PROBES_BED are `mutually exclusive` and `mandatory`, one or the other must be specified!

3. `PROBE_TARGETS: “{name of the BED containing probe targeted regions}”` - BED file containing coordinates of the exon coordinates (targeted by the probes or all exons). For testing purposes, the BED file with regions targeted by xGen Exome Research Panel v2 probes is already placed in the `input_data_wes`.

4. `MATCHING_LENGTH: “{integer}”` - specify the minimum matching length for the overlapping region between the probe sequence and the fragment sequence. The matching length determines the selection of fragments.
   Note 1: Before setting the parameter, investigate the length of the probes! Setting the matching length higher than any probe length will result in no fragments being selected for sequencing, further resulting in the simulation crashing.

f. WES- Simplified PCR

In addition to probe-capture enrichment, WES encompasses a simplified PCR simulation depending on only three parameters.

1. `NUM_CYCLES: "{integer}"`- specifies the number of PCR cycles to be performed. The parameter is `mandatory`.

   ADVANCED PARAMETERS

2. `K_PARAMETER_pcr: "{non-zero float}"`- k-parameter and midpoint cycle (next parameter) are responsible for the amplification efficiency drop during the PCR cycling modelled by the inverted sigmoid function. The k-parameter shapes the steepness of the drop. For more details check the chapter Theory and Processes behind the Simulation. The parameter is `optional`, the default value is 1.

3. `MIDPOINT_CYCLE: "{integer}"`- This parameter represents the cycle at which the amplification efficiency is 50%.
   Note 1: As PCR in WES is short, it is expected that the efficiency will be around 100% during all cycles. To achieve this, the midpoint cycle is set to 20, ensuring only a small variation of around 100% during 8 PCR cycles.

g. Sequencing parameters

This parameter set controls the sequencing, they are identical for both amplicon sequencing and WES simulation. Based on the number of post-PCR fragments and/or their lengths, the number of reads each fragment will get will be calculated and produced. The InSilicoSeq tool (v1.6.0) is the read-generation engine modified to produce reads from selected fragments.

1. `NUM_READS: "{integer}"`- specifies the number of reads to be produced. The parameter is `mandatory`.
   Note 1: Since the number of read pairs is calculated based on the multinomial distribution (read the Theory and Processes behind the Simulation), the actual number of reads produced might differ to a certain extent from the given number.

2. MODE: "{sequencing mode}" - specifies the sequencing mode. It can be "perfect" or "basic" indicating that the error model would not be used, and produced reads would be without any errors; "kde" specifies the usage of the kernel density estimator model that uses specific ERROR_MODEL (next parameter). The parameter is `optional`, if not specified it will use the "kde" mode as the default.

3. `ERROR_MODEL: "{sequencing error model}"`- specifies the error model for the specific Illumina instrument. This is the unchanged parameter from the InSilicoSeq tool which offers several error models for different illumina instruments "novaseq", "miseq", "hiseq", and "nextseq". For more details about the error models and how to build one, please refer to the InSilicoSeq documentation. The parameter is `optional`.

Note1: To generate error-free reads, ensure the ERROR_MODEL parameter is unset and configure the MODEL parameter to either "perfect" or "basic." Conversely, to simulate sequencing errors characteristic of a specific Illumina instrument, leave the MODEL parameter unset and specify the desired instrument in the ERROR_MODEL parameter.

4. GC_BIAS: "{--gc_bias}" - specifies whether the GC bias should be applied, another InSilicoSeq build-in feature. Reads with abnormal GC content will not be produced. The bias directly influences the number of reads specified to be produced. The parameter is `optional`, leave the string empty ("") if the bias should be omitted.
5. `FASTA_GZ_OUTPUT: "{--compress}"`-specifies whether the fastq output should be compressed or not. The parameter is `optional`, leave the string empty ("") if compression should be omitted.

6. `OPT_FRAG_LENGTH: "{specific mode}"` specifies the mode for read calculation that alters the sequencing probability based on the fragment length. The probability is thereby skewed towards the shorter fragments that will be favoured for sequencing. The mode operates together with the K_PARAMETER_seq (advanced parameter described under). The parameter is `mandatory`. For a detailed explanation please read the chapter Theory and Processes behind the Simulation. Several modes can be used:
    a) NO - the mode will not be used; the read calculation is based solely on the abundance of the fragments.
    b) `fixed {integer}` - Fragments with the specified length will have a 50% chance of being sequenced. Fragments having lengths lower than the specified will have an increased probability for sequencing, than fragments with lengths higher than the specified length.
    c) `default` - similar to `fixed` mode, where the specified fixed length is 350.
    d) `median` - the median of all fragment lengths is identified, fragments with lengths lower than this value will have an increased chance of being sequenced, than longer fragments.
    e) `quartile {1|2|3|4}` - identifies the specified quartile value based on all fragment lengths. Fragments with a length lower than this value will have an increased chance of being sequenced, than longer fragments.

ADVANCED PARAMETERS

7. `K_PARAMETER_seq: "{non-negative non-zero float}"` - The parameter is necessary only when OPT_FRAG_LENGTH is other than "NO".
Note 1: The parameter is `mandatory` when OPT_FRAG_LENGTH is other than NO, and we recommend a 0.05 value when used.

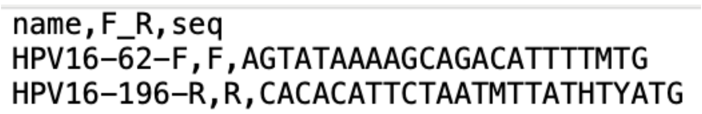
IV) Input files

In this chapter, we will cover the necessary and optional input files. Be aware of the “Notes”, they will usually be more in the form of dos and don’ts. All input data must be placed in the input_data folder. Certain input files are amplicon sequencing or WES-specific, if not specified file can be used in both simulations.

a. Necessary input:

1. ***FASTA or FASTA.gz file*** with the genome sequence(s) - the simulation allows the usage of different genomes/chromosomes stored in one FASTA file.
   Note1: use as few as possible characters for the genome name in the header of the FASAT file - headers are used as identifiers in the outputs, the fewer characters, the less memory is required for their storage.

2. ***CSV file with primer names and their sequences*** required for **amplicon sequencing**- by default, play dataset uses `all_primers.csv` file. This is the whole set of primers used for the whole genome sequencing of HPV16 in the TaME-seq method (Table 1). The format of the comma-separated table must be the same. The “name” column contains primer names, the “F_R” column contains information about the primer orientation, and “seq” is the primer sequence.
   Note 1: The primer name has to contain an -F or -R suffix depending on the orientation of the primer.
   Note 2: Ambiguous bases are allowed in the primer sequences - and they will be used as such (more details can be found in the Theory and Processes behind the Simulation)
   Note 3: None of the input values in the table can contain blank spaces.
   Note 5: GENOMICON-seq allows the simulation of many PCR reactions, and this file has to include ALL the primers used in the simulation, even if they will be used in different PCR simulations.

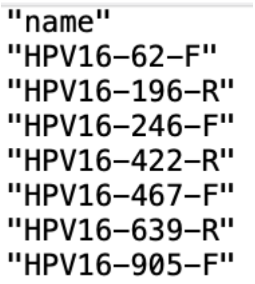


```
name,F_R,seq
HPV16-62-F,F,AGTATAAAAGCAGACATTTTMTG
HPV16-196-R,R,CACACATTCTAATMTTATHTYATG
```

Table 1. Example all_primer.csv table

3. ***CSV file with primer names***, one file per PCR reaction required for **amplicon sequencing** - Depending on the specified number of PCR reactions, for each reaction, a separate primer set has to be pre-made. In our input_data_ampliseq from the play dataset, we have 2 primer_set CSV files, `primer_set_1.csv` and `primer_set_2.csv`, the index number at the end specifies in which PCR reaction, primers should be used. This is a single-column table (column “name”) with ONLY primer names (Table 2). In case of only one PCR reaction, all names stored in the all_primers.csv should also be in the primer_set_1.csv.

Note 1: Make as many primer_set CSV tables as the number of specified PCR reactions.
Note 2: The base name of primer_set_index.csv, must not be changed, only the index number should correspond to the PCR reaction number primers will be used.
Note 4: Primer names in this table must be identical to the primer names in the all_primer.csv table.
Note 5: Ensure that the table does not contain any blank spaces.

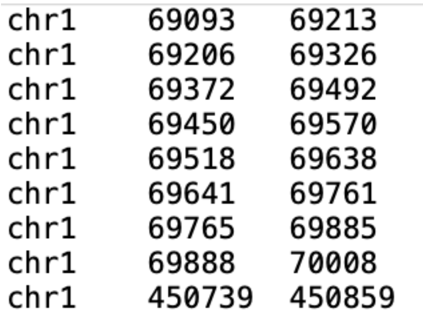


```
"name"
"HPV16-62-F"
"HPV16-196-R"
"HPV16-246-F"
"HPV16-422-R"
"HPV16-467-F"
"HPV16-639-R"
"HPV16-905-F"
```

Table 2. Example of the table primer_set_1.csv

4. ***FASTA or BED probe file*** - required for **WES** -Either one of the files needs to be specified. For a BED file, it is required to have three columns, chromosome name, and start and end coordinates of the probe binding sites (Table 3).
Note 1: Remove additional columns in the BED file as they will not be required (if the specified BED file contains additional columns).
Note 2: In our example, we are using xGen Exome Research Panel v2 probes. If the probe file is unavailable, the BED file containing exonic regions can also be used.

5. ***BED file with exon regions targeted by probes*** - required for **WES** - The file is required for mutation generation. Generated mutations will be found only in the regions specified by this file (Table 3).



```
chr1    69093    69213
chr1    69206    69326
chr1    69372    69492
chr1    69450    69570
chr1    69518    69638
chr1    69641    69761
chr1    69765    69885
chr1    69888    70008
chr1    450739   450859
```

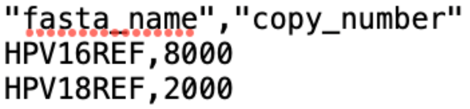
Table 3. Example of the BED file format for both probes and their targets.

6. ***SQL database*** - Besides the input files in the main input directory, the simulation requires an SQL database. However, the SQL database can either be made during the simulation or downloaded. Structurally the SQL database is a table,

containing 6 columns, nucleotide, comp_nucleotide (complementary nucleotide), tri_context (trinucleotide context), penta_context (pentanucleotide context), tri_contex_rev_comp (reverse complementary trinucleotide context), and penta_context_rev_comp (reverse complementary pentanucleotide context). For each position in a sequence, the database has information about which nucleotide is in this position, what is its complement, and what trinucleotide and pentanucleotide context both nucleotides at this position and its respective complement are found in. A separate database is created for each sequence in the multifasta file. This enables a quick search and retrieval of nucleotides and their specified contexts, as well as sequences required during the simulation. If not found, the database will be created only once for each sequence in the fasta file, and then reused in later runs as long the sequence header is unchanged.

b. Optional input:

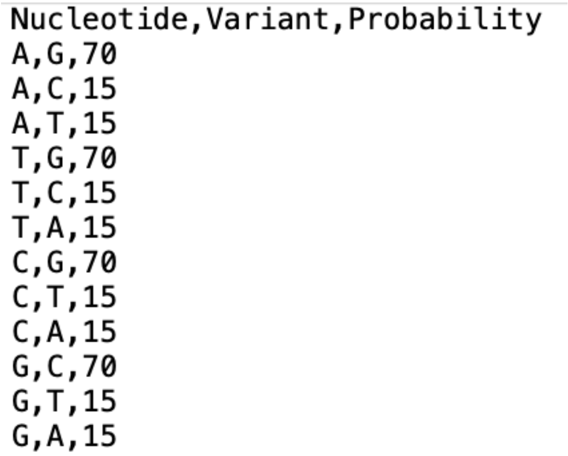
1. ***CSV table with the copy number of different genomes*** (`multifasta_copy_number.csv` in the `input_data` folder) - In the case where your FASTA file contains many different genomes/chromosomes, their individual copy numbers can be specified in this table- otherwise, parameter `NR_COPIES` is applied on all genomes/chromosomes in the FASTA file. The table has two columns, “fasta_name” and “copy_number” (Table 4), where the fasta name is the header of each fasta sequence in the FASTA file, while “copy_number” is the number of copies for each of the sequences.
Note 1: Do not change the format of the table.
Note 2: Names of the fasta sequences must be identical to the headers of sequences in the FASTA file.



```
"fasta_name","copy_number"
HPV16REF,8000
HPV18REF,2000
```

Table 4. Example of the multifasta_copy_number.csv

2. ***CSV table with the probability of each nucleotide being substituted with other nucleotides*** - this table replaces the parameter `TS_TV_RATIO` (transitions vs transversions ratio). It enables the specification of the mutation probability of each nucleotide to mutate to any other nucleotide but itself (Table 5). The table has 3 columns, “Nucleotide”, “Variant” and “Probability”, where for each nucleotide the probability to be mutated to any of the three variants is specified.
Note 1: The probability is given in %.
Note 2: All combinations have to be specified!
Note 3: Probability can be set to 0 for the specific change of one nucleotide to the other.
Note 4: Blank spaces must be removed!

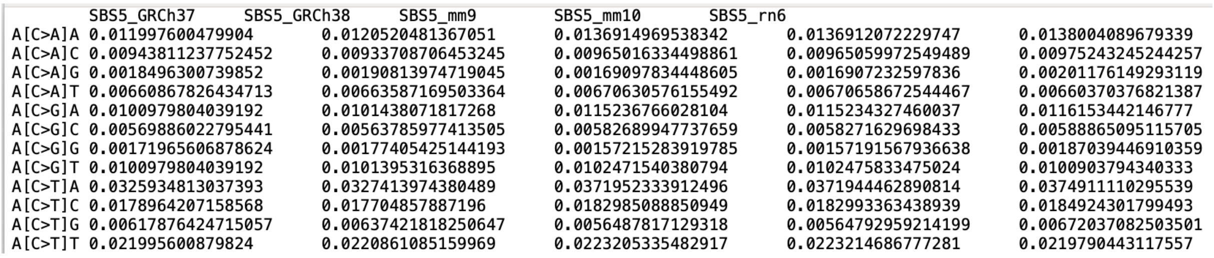


```
Nucleotide,Variant,Probability
A,G,70
A,C,15
A,T,15
T,G,70
T,C,15
T,A,15
C,G,70
C,T,15
C,A,15
G,C,70
G,T,15
G,A,15
```

Table 5. Example of the probability_variant_table.csv

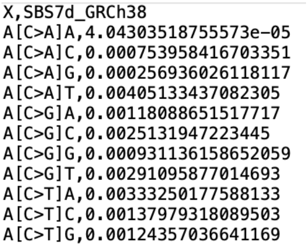
3. ***SBS table containing the percentage of single base substitutions (SBS) required only in WES*** - For each specific SBS signature, a table in numeric form can be downloaded and trimmed to contain only the required information from the COSMIC database. The tables contain the SBS percentage for hg37, hg38, mm9, mm10, and rn6. Please use the `sbs_table_trimming.py` that can be found here. That converts the original SBS-signature table (Table 6A) to the simulation-required format (Table 6B). Download the SBS-signature table and save it as a csv file, for example, SBS2_table.csv. Each table has values for each 96 mutational contexts stored in 5 columns `SBS2_GRCh37`, `SBS2_GRCh38`, `SBS2_mm9`, `SBS2_mm10`, `SBS2_rn6`, pick the column name (for hg37 or hg38), we used SBS2_GRCh38, and run the script python `sbs_table_trimming.py --input_sbs path/the/table/SBS2.csv --column_name SBS2_GRCh38 --output_sbs path/to/output/SBS2_selected.csv`
   Place the generated table into the `input_data_wes` directory and set up a table name in the `config_wes.yml`.

A)



```
        SBS5_GRCh37     SBS5_GRCh38     SBS5_mm9        SBS5_mm10       SBS5_rn6
A[C>A]A 0.011997600479904       0.0120520481367051      0.0136914969538342      0.0136912072229747      0.0138004089679339
A[C>A]C 0.00943811237752452     0.00933708706453245     0.00965016334498861     0.00965059972549489     0.00975243245244257
A[C>A]G 0.0018496300739852      0.00190813974719045     0.00169097834448605     0.0016907232597836      0.00201176149293119
A[C>A]T 0.00660867826434713     0.00663587169503364     0.00670630576155492     0.00670658672544467     0.00660370376821387
A[C>G]A 0.0100979804039192      0.0101438071817268      0.0115236766028104      0.0115234327460037      0.0116153442146777
A[C>G]C 0.00569886022795441     0.00563785977413505     0.00582689947737659     0.0058271629698433      0.00588865095115705
A[C>G]G 0.00171965606878624     0.00177405425144193     0.00157215283919785     0.00157191567936638     0.00187039446910359
A[C>G]T 0.0100979804039192      0.0101395316368895      0.0102471540380794      0.0102475833475024      0.0100903794340333
A[C>T]A 0.0325934813037393      0.0327413974380489      0.0371952333912496      0.0371944462890814      0.0374911110295539
A[C>T]C 0.0178964207158568      0.017704857887196       0.0182985088850949      0.0182993363438939      0.0184924301799493
A[C>T]G 0.00617876424715057     0.00637421818250647     0.0056487817129318      0.00564792959214199     0.00672037082503501
A[C>T]T 0.021995600879824       0.0220861085159969      0.0223205335482917      0.0223214686777281      0.0219790443117557
```

B)



```
X,SBS7d_GRCh38
A[C>A]A,4.04303518755573e-05
A[C>A]C,0.000753958416703351
A[C>A]G,0.000256936026118117
A[C>A]T,0.00405133437082305
A[C>G]A,0.00118088651517717
A[C>G]C,0.0025131947223445
A[C>G]G,0.000931136158652059
A[C>G]T,0.00291095877014693
A[C>T]A,0.00333250177588133
A[C>T]C,0.00137979318089503
A[C>T]G,0.00124357036641169
```

Table 6. A) SBS signature table before trimming and conversion, B) SBS signature table after trimming and conversion to csv format.

V) Output files and how they may serve you

The main outputs are paired R1 and R2 FASTQ files. However, as GENOMICON-seq modulates the mutation-to-noise ratio, tracking down the inserted mutations and error mutations requires additional files. We will go through each output file in the order they are produced during both amplicon sequencing and WES simulations. Many of the outputs are identical while some are specific for **amplicon sequencing** or **WES** simulation.

A) Amplicon sequencing simulation output directory structure

```
Output_data_ampliseq
└── sample_1
    ├── generated_reads
    ├── log_folder
    └── tech_replicate_1
        ├── PCR
        │   ├── PCR_1
        │   └── PCR_2
        └── log_folder
```

B) WES simulation output structure

```
Output_data_wes
└── sample_1
    ├── generated_reads
    ├── log_folder
    └── tech_replicate_1
        ├── PCR_reaction
        └── log_folder
            └── PCR_reaction
```

Figure 1. Folder structure of A) Amplicon sequencing simulation, and B) WES simulation

Output_data_ampliseq or Output_data_wes are the specified folders containing all outputs in our example (Figure 1). For the specified number of samples to be generated, sample folders will be produced designated as sample_{index}, index=1,2,3...
generated_reads will contain the fastq files for each technical replicate (and each PCR reaction simulated in **amplicon sequencing**). log_folder will contain log files from sample generation.

For the specified number of technical replicates, folders named technical_replicate_{index}, index = 1,2,3,..., will be produced. The folder will contain files with the fragment coordinates for each fragmented technical replicate. log_folder contains the log file from the fragmentation process.

In **amplicon sequencing simulation**, the PCR folder contains the prep files for each specified PCR reaction. Folders PCR_index, index =1,2,3,... will be produced for each specified PCR reaction. The folder holds all files necessary for read generation. In the **WES simulation**, the PCR_reaction folder contains all the necessary files for the read-generation process.

a. Output_data folder content

The files in this folder are identical for both amplicon sequencing and WES (Figure 2). Besides the sample_{index} folder, the seed_log.txt. contains the major seed and all other seeds derived from the major seed. The major seed controls the generation of all seeds and might come in handy when the simulated data should be reproduced.

i. sample_{index} folder

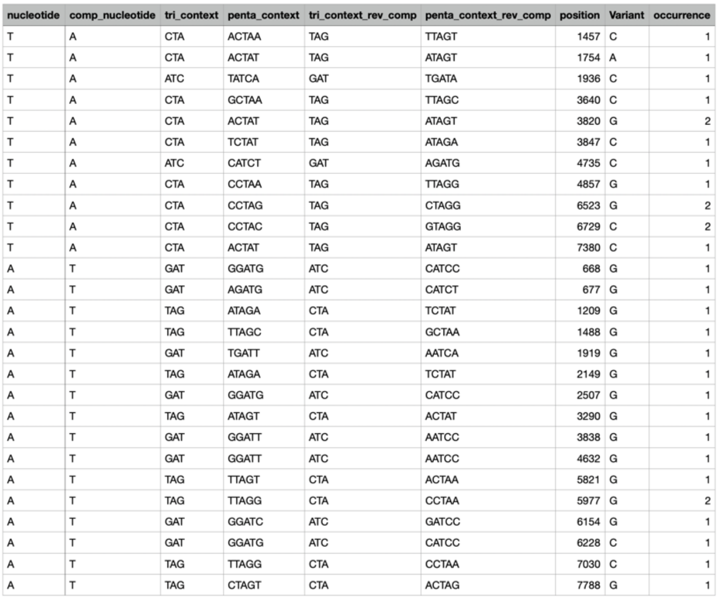
```
sample_1
├── cleanup_1_complete.txt
├── genome1_inserted_mutations_overview.csv
├── genome1_mutation_counts.csv
├── fasta_lengths.csv
├── generated_reads
├── log_folder
└── tech_replicate_1
```

Figure 2. Overview of the sample_index folder content. Content is identical for both amplicon sequencing and WES simulation with the exception of fasta_lengths.csv which is produced only in amplicon sequencing simulation.

The simulation starts by generating an in-silico sample with a specified number of genome copies and mutations based on the specifications. Sample generation output files are placed in this directory.

For each sequence in the fasta file, `inserted_mutations_overview.csv` and one `mutation_counts` will be produced with the sequence name as the prefix. For example, if the fasta file contains genome_1 and genome_2 sequences, the corresponding csv file names will be `genome_1_sample_table.csv` and `genome_2_sample_table.csv`, and `genome_1_mutations_table.csv` and `genome_2_mutations_table.csv`.

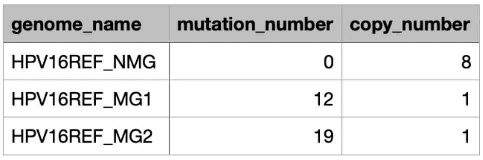
- `inserted_mutations_overview.csv` - holds the information about all inserted mutations, their positions, the original nucleotide at that position, its complement nucleotide, and tri and pentanucleotide contexts for both nucleotide and their complements, the nucleotide replacing the original one, and in how many original genome/chromosome copies the nucleotide has been replaced (column occurrence).



nucleotide	comp_nucleotide	tri_context	penta_context	tri_context_rev_comp	penta_context_rev_comp	position	Variant	occurrence
T	A	CTA	ACTAA	TAG	TTAGT	1457	C	1
T	A	CTA	ACTAT	TAG	ATAGT	1754	A	1
T	A	ATC	TATCA	GAT	TGATA	1936	C	1
T	A	CTA	GCTAA	TAG	TTAGC	3640	C	1
T	A	CTA	ACTAT	TAG	ATAGT	3820	G	2
T	A	CTA	TCTAT	TAG	ATAGA	3847	C	1
T	A	ATC	CATCT	GAT	AGATG	4735	C	1
T	A	CTA	CCTAA	TAG	TTAGG	4857	G	1
T	A	CTA	CCTAG	TAG	CTAGG	6523	G	2
T	A	CTA	CCTAC	TAG	GTAGG	6729	C	2
T	A	CTA	ACTAT	TAG	ATAGT	7380	C	1
A	T	GAT	GGATG	ATC	CATCC	668	G	1
A	T	GAT	AGATG	ATC	CATCT	677	G	1
A	T	TAG	ATAGA	CTA	TCTAT	1209	G	1
A	T	TAG	TTAGC	CTA	GCTAA	1488	G	1
A	T	GAT	TGATT	ATC	AATCA	1919	G	1
A	T	TAG	ATAGA	CTA	TCTAT	2149	G	1
A	T	GAT	GGATG	ATC	CATCC	2507	G	1
A	T	TAG	ATAGT	CTA	ACTAT	3290	G	1
A	T	GAT	GGATT	ATC	AATCC	3838	G	1
A	T	GAT	GGATT	ATC	AATCC	4632	G	1
A	T	TAG	TTAGT	CTA	ACTAA	5821	G	1
A	T	TAG	TTAGG	CTA	CCTAA	5977	G	2
A	T	GAT	GGATC	ATC	GATCC	6154	G	1
A	T	GAT	GGATG	ATC	CATCC	6228	C	1
A	T	TAG	TTAGG	CTA	CCTAA	7030	C	1
A	T	TAG	CTAGT	CTA	ACTAG	7788	G	1

Table 6. Example of the `inserted_mutations_overview.csv`

- `mutation_counts.csv` - holds the information about how many genome copies are mutated, and how many mutations they harbour (Table 7). All non-mutated genome copies have an _NMG identifier (non-mutated genome), while all mutated genome copies have _MG* identifier, where the * specifies the current number of uniquely mutated copies.



genome_name	mutation_number	copy_number
HPV16REF_NMG	0	8
HPV16REF_MG1	12	1
HPV16REF_MG2	19	1

Table 7. Example of the `mutations_counts.csv`

If the mutation rate is set to 0, or if a very low mutation rate is specified, no mutations will be inserted. The `inserted_mutations_overview.csv` will not be produced, and the `mutation_counts.csv` will only list the number of copies for the non-mutated genomes.

- `fasta_lengths.csv` - needed by the simulation (**amplicon sequencing** simulation) listing the lengths of all genomes in the FASTA. For **WES** simulation the file is produced and stored in the input_data. Does not contain any user-useful information.

- log_folder - for each sample folder, a log folder will be produced containing the log files `Making_SQL_database.log`, and `Defining_seq_and_mut.log`. `Making_SQL_database.log` will indicate which SQL databases were produced if their production was necessary. `Defining_seq_and_mut.log` will print out the parameters used for sample generation and the mutational process you specified.

- `cleanup_1_complete.txt` - an empty text file signalizing that the first cleanup process is finished, does not contain any user-useful information

- `generated_reads` folder is produced at the end of the simulation and contains the R1 and R2 FASTQ.gz files for each technical replicate of the given sample.

ii. tech_replicate_{index} folder

For each technical replicate, one folder will be generated. The content of this folder is slightly different between the two simulations (Figure 3).

A) Amplicon sequencing

```
tech_replicate_1
├── genome1_fragments.bed.gz
├── PCR
└── log_folder
```

B) WES

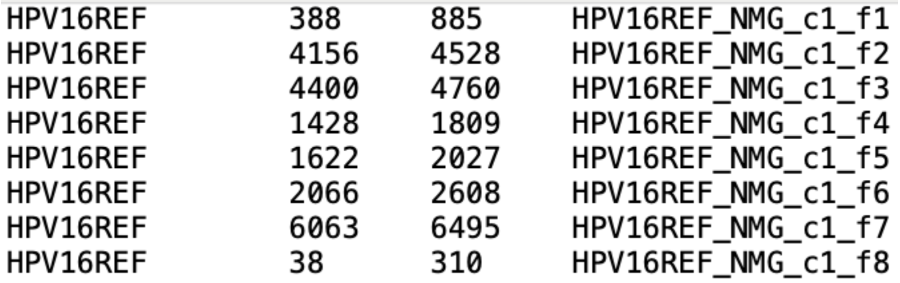
```
tech_replicate_1
├── PCR_filtered.bed.gz
├── PCR_reaction
└── log_folder
```

Figure 3. The overview of the tech_replicate_index folder content in A) amplicon sequencing, and B) WES simulation.

1. Amplicon sequencing simulation

In amplicon sequencing the folder contains the results from the fragmentation process for each specific fragmentation replicate. Each sequence in the FASTA file will get its own set of outputs with their headers placed as a prefix in the file names.

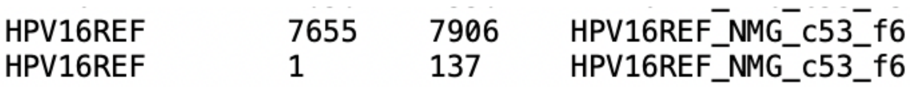
- `fragments.bed.gz` - BED file containing the coordinates of all the fragments produced during fragmentation. Table 9 represents an example of how this bed file looks like. The 1st column is the name of the FASTA sequence used (note that each fasta sequence if multifasta is used will still have its own file), the 2nd and 3rd columns are the start and end coordinates of the fragment, respectively, and the 4th column is the fragment name. Each fragment has a unique identifier with a suffix `_c{index}` indicating from which NMG or MG copy the fragment originates, and `_f{index}` indicating the index number of a fragment. This file will be split into the specified number of PCR reactions.



HPV16REF	388	885	HPV16REF_NMG_c1_f1
HPV16REF	4156	4528	HPV16REF_NMG_c1_f2
HPV16REF	4400	4760	HPV16REF_NMG_c1_f3
HPV16REF	1428	1809	HPV16REF_NMG_c1_f4
HPV16REF	1622	2027	HPV16REF_NMG_c1_f5
HPV16REF	2066	2608	HPV16REF_NMG_c1_f6
HPV16REF	6063	6495	HPV16REF_NMG_c1_f7
HPV16REF	38	310	HPV16REF_NMG_c1_f8

Table 9. Example of the fragments.bed.gz

Note 1: If the genome is defined as **`circular`**, a fragment can be split between the end of the genome and the genome start. The fragment record is then written in two rows with the same fragment identifier (example in Table 10).



HPV16REF	7655	7906	HPV16REF_NMG_c53_f6
HPV16REF	1	137	HPV16REF_NMG_c53_f6

Table 10. Example of the split fragment due to the circular nature of the genome. The identifier is the same, however, fragment coordinates go from 7655 until 7906, which is the end of the HPV genome, and from 1 until position 137.

2. WES simulation

The only output file in this folder in the WES simulation is the `PCR_filtered.bed.gz`. The file contains the coordinates of all fragments which matched the probes' binding coordinates with the minimum specified matching length. Strictly speaking, the file contains the fragments enriched by probes and is now ready for the short PCR reaction simulation.

iii. PCR and PCR_reaction folders in amplicon sequencing and WES simulations

Due to differences in the complexity of the simulated PCRs between amplicon sequencing and WES, the folder structure and content are also different (Figure 4).

A) Amplicon sequencing simulation

```
PCR
├── genome1_splitting_complete.log
├── PCR_1
├── PCR_1.log
├── PCR_2
├── PCR_2.log
├── fragments_pcr1.bed.gz
├── fragments_pcr2.bed.gz
└── merging_complete.log
```

B) WES simulation

```
PCR_reaction
├── chr1_seq_mutations_overview.csv
└── sequenced_frags.fasta.gz
```

Figure 4. The overview of the PCR and PCR_reaction folder contents in A) amplicon sequencing, and B) WES simulation, respectively.

1. Amplicon sequencing simulation - PCR folder

In the PCR folder, each PCR reaction will have its own folder (Figure 4A). In this example, where 2 PCR reactions were specified there will be two PCR folders. In addition, each reaction will have its own `fragments_pcr{index}.bed.gz`. Until now, if a multifasta file was used, each of the given genome sequences was fragmented separately. Before PCR simulation, fragments from different genomes (stored in `{genome_name}_fragments.bed.gz` in `tech_replicate` folder) will be combined so that the PCR can be performed on all the fragments selected for the PCR reaction regardless of the genome they are originating from if the provided primers bind to these fragments. Combined fragments are then split into the number of files corresponding to the number of specified PCR reactions.

- `fragments_pcr{index}.bed.gz` - For each PCR reaction there will be one `fragments_pcr{index}.bed.gz`. The structure of the file is identical to `{genome_name}_fragments.bed.gz`, providing details on which fragment underwent processing in each PCR reaction.

- `merging_complete.log` and `splitting_complete.log` - are empty log files, necessary for signaling the pipeline to continue.

- PCR_`index.log` file lists the parameters used in the PCR reaction in addition to a set of seeds used for each process.

2. *Amplicon sequencing - PCR_{index} folder*

```
PCR_1
├── genome1_seq_mutations_overview.csv
├── genome2_seq_mutations_overview.csv
├── PCR_1-polymerase_error_mutations.csv
└── PCR_1_sequenced_frags.fasta.gz
```

Figure 5. The overview of the PCR_{index} folder content.

For each performed PCR simulation, several files will be produced (Figure 5):

- `PCR_{index}_sequenced_frags.fasta.gz` - Sequences of all unique amplicons that got sequenced are produced and stored in the fasta.gz file. The sequences contain mutations inserted during the sample generation and during the PCR.

- `PCR_{index}-polymerase_error_mutations.csv` lists all polymerase error mutations inserted during the reaction. The 1st column indicates mutation position in regards to the reference genome and a nucleotide that substituted the original one. Error mutation might also replace the mutation inserted during the sample generation. The second column is the occurrence of the error mutations on different fragments. If the simulation was PCR-error-free, the log file stating that no mutations were inserted during the PCR will be produced.

- `{genome_name}_seq_mutations_overview.csv` - Each genome in the multifasta file will get its own `_seq_mutations_overview.csv`. The file has an identical structure as the `inserted_mutations_overview.csv` in `sample_{index}` folder, now showing the mutations on fragments selected for sequencing (in the `sequenced_frags.fasta.gz`) and their occurrences. If the simulation was free from inserted mutations, or if no mutations have been selected to be sequenced, the file will be replaced with a log file stating that no inserted mutations have been sequenced.
  Note1: The mutations can be lost during the simulated processes, thereby the numbers between `inserted_mutations_overview.csv` and `sequenced_frags.fasta.gz` can differ in both the number of mutations and their occurrence. In addition, the occurrence represents the number of each mutation found on a unique fragment that will be sequenced, not the number of reads holding the mutation.

3. *PCR_reaction folder in the WES simulation*

Since WES simulation encompasses only one simple PCR reaction, this folder will contain all the outputs and are almost identical to filers in amplicon sequencing stored in the `PCR_{index} folder` (Figure 4B).

- `chr1_seq_mutations_overview.csv` - Each genome in the multifasta file will get its own `_seq_mutations_overview.csv`. The file has an identical structure

as the `inserted_mutations_overview.csv` in `sample_{index}` folder, now showing the mutations on fragments selected for sequencing (in the `sequenced_frags.fasta.gz`) and their occurrences. If the simulation was free from inserted mutations, or if no mutations have been selected to be sequenced, the file will be replaced with a log file stating that no inserted mutations have been sequenced.
Note1: The mutations can be lost during the simulated processes, thereby the numbers between `inserted_mutations_overview.csv` and `sequenced_frags.fasta.gz` can differ in both the number of mutations and their occurrence. In addition, the occurrence represents the number of each mutation found on a unique fragment that will be sequenced, not the number of reads holding the mutation.

- `sequenced_frags.fasta.gz` contains all the unique sequences of fragments that got sequenced in a FASTA.gz format. The sequences contain the mutations inserted during the sample generation process.

iv. generated_reads folder

Finally, all generated R1 and R2 FASTQ.gz files for each specified replicate would be found in a generated_reads folder placed in a `sample_{index}` folder (Figure 5).

```
generated_reads
├── tech_replicate_1_PCR_1_R1.fastq.gz
├── tech_replicate_1_PCR_1_R2.fastq.gz
├── tech_replicate_1_PCR_2_R1.fastq.gz
└── tech_replicate_1_PCR_2_R2.fastq.gz
```

Figure 5. Content of the generated_reads folder.

The most important files for tracking mutations are `inserted_mutations_overview.csv and _seq_mutations_overview.csv`, and for amplicon sequencing PCR_`{index}-polymerase_error_mutations.csv`. Unfortunately, until now, the sequencing errors are not recorded anywhere.

After the read-mapping, alternative alleles and how many reads support them can be identified. Information from the `inserted_mutations_overview.csv and seq_mutations_overview.csv` can be used to determine how many mutations have been lost during the simulated library prep process, and how the frequency of mutations changes. While in WES it can be expected that the mutation frequencies will be lower before and after the simulated library prep processes, in the amplicon seq the frequency of certain mutations might rise. In amplicon sequencing, the number of unique fragments increases with the number of PCR errors, which also increases the frequency of mutations inserted during the sample generation. In addition, in amplicon sequencing, PCR_`{index}-polymerase_error_mutations.csv` can also assist in determining which detected alternative alleles originate from the polymerase error.

For the in-depth analysis, the reads could be split into several sub-FASTAQ files. Each read has its identifier specifying the fragment origin, mutated genome (MG identifier) or non-mutated genome (NMG identifier), and specifying positions inserted during the PCR in the case of amplicon sequencing simulation. If no PCR errors were inserted a read will have "_No_mutations_" as part of its name (in WES, all reads have "_No_mutations_" as polymerase bias was omitted). Mapping the reads independently might further uncover whether the signal from the true mutations got modified by sequencing in WES simulation, or by sequencing and PCR in amplicon sequencing.

VI) The Theory and Processes Behind the Simulation

In this chapter, we will introduce the details of the GENOMICON design in terms of the different models used and how each of them affects the final results.

a. Sample generation and insertion of mutations

The ideal sample would contain a specified copy number of selected genomes/ chromosomes. The sample would then be mutated by user-customizable mutation insertion mode. The deterministic mode operates with a specified fraction of genomes/ chromosomes that will be mutated, and a specified fraction of positions that will be mutated. Specific mutation rate mode requires only the customizable mutation rate. SBS-mimicry mode is reserved exclusively for WES and inserts the mutations whose distribution will resemble the specified SBS mutational profile.

Scripts and Outputs:

- **Amplicon Sequencing**: The `mut_genomes.R` script is designed for amplicon sequencing simulation. It efficiently processes each specified genome sequence in parallel, accommodating the typically smaller size of genomes in such studies. In addition, the script requires an additional parameter specifying genome topology. The script generates two primary outputs: `sample_table.csv` and `mutations_table.csv` which will be transformed into more friendly outputs `inserted_mutations_overview.csv` and `mutation_counts.csv` in the final steps of the pipeline.
- **WES**: The `1mut_genomes.R` script functions similarly by producing the same output files. To enhance processing speed for the larger human genome, it handles each chromosome in the FASTA file sequentially. However, the mutation generation and output writing processes are executed in parallel to optimize performance and manage large-scale data efficiently. As the human genome's topology is constant, the script omits the specification of the genome topology but does require the specification of a file containing exonic regions.

i. Deterministic mode - Fraction of mutated genomes + fraction of positions

In deterministic mode, the tool calculates the exact number of genome copies to mutate by applying the specified fraction to the total number of genomes. Similarly, the exact number of mutation positions is determined by applying the fraction of positions to the total number of positions across the entire genome or within specific nucleotide contexts, if provided. For targeted nucleotide contexts, the calculation considers only the occurrences of those contexts to determine the available positions for mutation.
In deterministic mode, once the mutation positions are identified across genome copies, the type of nucleotide substitution is determined by the specified parameters: either `TS_TV_RATIO` or `VARIANT_PROBABILITY_TABLE`. These parameters define the likelihood of each potential nucleotide replacing the original at a given position. If no parameters are specified, any nucleotide other than the original can be randomly selected for substitution.

To introduce variability, not all selected genomes receive every mutation. Instead, the chosen mutations are randomly distributed among the selected genomes, ensuring that each genome receives at least one mutation. This method affects the overall distribution of mutations within the sample. Although the occurrence of each mutation will be roughly uniform across the genomes, the actual number of mutations and their specific combinations will vary among different genome copies. While you can specify the total number of mutations and genomes, the precise distribution of mutations remains beyond complete control.

ii. Specific mutation rate mode

This mode allows the user to simulate the mutation insertion using a simplified version of Kimura's infinite sites model. While the original model includes several assumptions, we adapt it specifically for simulating mutation occurrence without attempting to infer true evolutionary pathways. Key assumptions in our simplified model include:

- ***Rare Mutations***: Mutations are considered rare events, with the probability of multiple mutations at the same site being negligible. This treats each mutation as a unique occurrence.
- ***Infinite Mutation Sites***: There are theoretically infinite possible mutation sites in the genome, allowing for endless new mutations without the complications of mutations reverting or recurring at the same site.

Under these assumptions, the Poisson distribution models the likelihood of mutations occurring within a given sequence length. This is represented mathematically as $P(k;\lambda)= \lambda^{k}e^{-\lambda}/ k!$, where P is the probability of observing k mutations, λ is the expected number of mutations (mutation rate multiplied by sequence length).

Thereby, the Poisson distribution predicts the number of mutations for each genome copy, using the total genome length and specified mutation rate. Mutation positions are selected randomly for each genome (without duplicates), either from a total pool of positions or from a pool of positions matching the specified mutational context (MUT_CONTEXT). Similar to the Deterministic mode, the type of nucleotide substitution at each mutation position is governed by the TS_TV_RATIO or VARIANT_PROBABILITY_TABLE parameters, which define the likelihood of each potential nucleotide replacing the original.

iii. SBS - mimicry mode

The SBS-mimicry mode enables realistic simulation of mutation patterns, tailored specifically for WES. Like in the Deterministic mode, the SBS-mimicry mode uses a user-defined fraction of genomes to determine the number of genomes that will undergo mutations and a position fraction to calculate the total number of mutations. The mode represents the extension of the deterministic mode. The total number of mutation positions is calculated based on a user-defined fraction of positions, which determines the overall number of mutations in the sample. This method ensures that while the specific mutation details are defined by the SBS signatures, the extent and distribution of these mutations are controlled by the user through the fractions of genomes and positions.

This mode utilizes a user-customizable SBS signature table, that can be downloaded from COSMIC, which details the percentage occurrence of single base substitutions across all trinucleotide contexts. These percentages dictate the likelihood of each mutation type occurring within the genome, ensuring that the simulation mirrors real-world mutation patterns observed in cancer genomics.

The process begins by formating the probabilities for each context to ensure that the sum for all mutations within a context equals one. The process preserves the empirical representations of the mutation patterns.

Following this initial setup, a contextual analysis is performed by querying human exome to identify occurrences of each trinucleotide context. The frequency of each context is then used to randomly select positions for mutation, aligning mutation occurrences with their likelihood based on the SBS data.

For each mutation position selected, the mutation applied is again determined directly from the SBS table, which specifies the exact nucleotide changes (e.g., C to T, G to A). This direct application of mutation types from the table eliminates the need for additional parameters like transition-transversion ratios, as the SBS signatures provide a comprehensive blueprint of the mutation landscape.

These mutations are then randomly distributed across the specified number of genome copies.

b. Fragmentation

The generated genome copies in the sample generation step are further fragmented (Figure 6). The likelihood of a genome copy entering fragmentation is determined by the `FRAGMENTATION_FRACTION` parameter, which sets the probability of selection using a binomial distribution. Each genome copy is equally likely to be selected. The number of times a sample is fragmented is set by `FRAG_REPLICATES`, introducing variability as each replicate could produce different fragments.

Scripts and outputs:

- **Amplicon Sequencing simulation** - In amplicon sequencing, the fragment selection is managed in R also handling the genome topology (`fragmentation.R`). Fragmentation simulation is executed by a custom C++ function called from R (`fragmentation.cpp`).
- **WES simulation** - Selection of genome copies for fragmentation also uses a binomial distribution in R (`2_fragmentation_prep_wes.R`), but the fragmentation itself is carried out directly in C++ optimized for better utilization of processing cores (`fragmenation_batches.cpp`).

Outputs of both scripts are BED files containing a set of coordinates for each fragment specifying the start and end positions of the original genome fragments originate from.

Fragmentation begins by selecting a fraction of genomes to process, with each genome handled individually. The first fragment's start position within the genome is randomly chosen, ensuring it does not exceed the genome's total length. The end of this fragment is then calculated from the start position and fragment length, with the coordinates of the start and end recorded.

Following the initial extraction of the first fragment, the genome is conceptually split into two segments: from the start of the genome to just before the extracted fragment's start, and from just after the extracted fragment's end to the end of the genome. Each segment is then fragmented further, repeating the process of start and end position selection, coordinate recording, and splitting. This recursive fragmentation continues until no further fragments can be extracted, either because the remaining sections are too small or the entire genome has been successfully fragmented. This method ensures that all fragment coordinates are accurately recorded, enabling efficient processing while minimizing memory usage by using only coordinates instead of full sequences.

For amplicon sequencing involving circular genomes, each genome copy is rearranged determining the genome copy's new start position. The process simulates the random opening of a circular genome before fragmentation.

c. PCR simulation - Amplicon Sequencing

PCR amplification is subject to a multitude of variables. Modelling all known variables is not only challenging due to their complex interactions but also computationally prohibitive. Given that the number of PCR copies increases exponentially, the computational resources required would also increase at a similar rate. To enhance efficiency, we have streamlined the simulation process. Here, we outline the principal assumptions and models that underpin our PCR simulation.

- ***Primer binding is perfect*** - The simulation is not made to test new primers, but to include the already validated ones. We assume that all primers bind perfectly to their targets. This simplification omits the calculation of the Tm of each primer and then models the change of binding specificity when primers' Tm differs slightly from the PCR reaction temperature, for each primer, for each cycle.

- ***PCR amplification efficiency is not perfect, it drops following the inverted sigmoid function***. The PCR amplification efficiency starts with nearly 100% efficiency leading to the exponential growth of the copies. However, depletion of reagents, accumulation of by-products, and polymerase degradation lead to the amplification efficiency drop. In the end, the production of new copies reaches the plateau.
- ***Binomial distribution governs how many copies will be produced in each cycle*** - the calculated efficiency for a current cycle represents the probability of how many new copies will be produced from the copies in the current cycle.
- ***Polymerase error rate inserts mutations following the simplified infinite site model*** - The model is identical to the model used to generate mutation in the original sample. The only difference is that Poisson distribution is used to calculate the number of mutations for each newly generated fragment copy in each cycle. In addition, every nucleotide in an amplicon can mutate to any other nucleotide with equal probability.
- ***More complex artefacts such as chimeric amplicons are not generated*** - in this simulation, the focus is on mutation detection and how it is influenced by PCR and sequencing errors. The integration of these complex by-products of PCR reaction is therefore omitted.
- ***Each fragment is amplified as a separate unit*** - while the amplification of each amplicon in a PCR reaction is not 100% independent from all the other amplicons, we simplified the process to gain speed.

PCR simulation involves two key stages: first, determining primer-binding sites on fragments to identify amplifiable regions; second, PCR cycling to amplify these regions (Figure 6).

i. Finding primer binding positions on each fragment.

All specified primers are mapped onto specified FASTA sequence(s) using the Bowtie2 (v2.4.4). This step determines binding sites for each primer on the FASTA sequence(s). Once mapping is complete, the results, initially in BAM format, are converted into a BED format using bedtools (v2.31.1), holding the records of the start and end positions of each primer's binding site.

For simulations involving multiple PCR reactions, the BED file containing primer coordinates is appropriately split. This ensures that each PCR simulation is equipped with only the relevant primer data it requires. Similarly, the initial pool of DNA fragments, also stored in a BED file, is randomly divided to match the number of specified PCR reactions. Consequently, each PCR reaction is paired with its own unique set of fragment and primer BED files.

The crucial step involves analyzing the overlap between the primer binding coordinates and the fragment coordinates pinpointing which parts of the fragments will be amplified. It's important to note that multiple F and R primers may bind to the same fragment. The amplifiable portion of each fragment is determined by the binding endpoint of an F primer and the starting point of an R primer. For every pair of F and R primers that binds to a fragment, their combined coordinates are recorded.

Finally, the process includes a filtering step where any identified amplifiable fragment regions that fall below the specified amplicon length (parameter `AMPLICON_LENGTH`) are discarded.

While primer binding is a critical component of the PCR reaction, simulating this step would be computationally demanding. In actual PCR procedures, shorter amplicons are often produced, but these are typically eliminated during the post-PCR size selection process, which retains only amplicons of the desired length. To enhance efficiency and avoid unnecessary computational expense, our simulation preemptively removes shorter amplicons before initiating the PCR simulation. This adjustment streamlines the process by focusing only on amplicons that are relevant for further analysis.

1. *Degenerate primers*

Amplicon sequencing might comprise degenerate primers whose sequence contains the ambiguous base. GENOMICON-Seq simulation accepts the primers with ambiguous bases in their sequence. It searches for the ambiguous bases then generates all the primer versions and uses all of them for mapping. Bowtie2 handles these small changes in the primer sequence quite well and records the mapping quality for all the primer versions. In the final step, only one set of coordinates is kept, the one with the best mapping length and highest quality.

While degenerate primers might lead to unspecific amplifications in real-life PCR, we are assuming that these will represent the minority of produced amplicons. Our method does not include the primer binding efficiency calculation but only handles the ambiguous primers as input.

ii. PCR cycling

1. *Inverted sigmoid function - a model for the amplification efficiency drop*

PCR amplification in real life typically progresses through three phases: an initial exponential phase where efficiency is highest, a linear phase where efficiency declines due to limited reagents, and a plateau phase where no further amplification occurs. To model these dynamics, we employ an inverted sigmoid function: $\frac{1}{1+e^{k\times(current\ cycle-midpoint\ cycle)}}$. This function effectively represents the gradual decrease in amplification efficiency, capturing the transition from exponential to plateau phases with an S-shaped curve.

The function is governed by two key parameters: the midpoint cycle and the k-parameter. The midpoint cycle defines the cycle number at which PCR efficiency is 50%, marking the transition point between the exponential and linear amplification phases. The k-parameter dictates the steepness of the efficiency decline; a higher k-value results in a sharper drop post-midpoint, simulating rapid reagent depletion. Conversely, a lower k-value indicates a slower decline in efficiency. Adjusting these parameters allows control over the amplification efficiency across different cycles, as depicted in Figure 6.

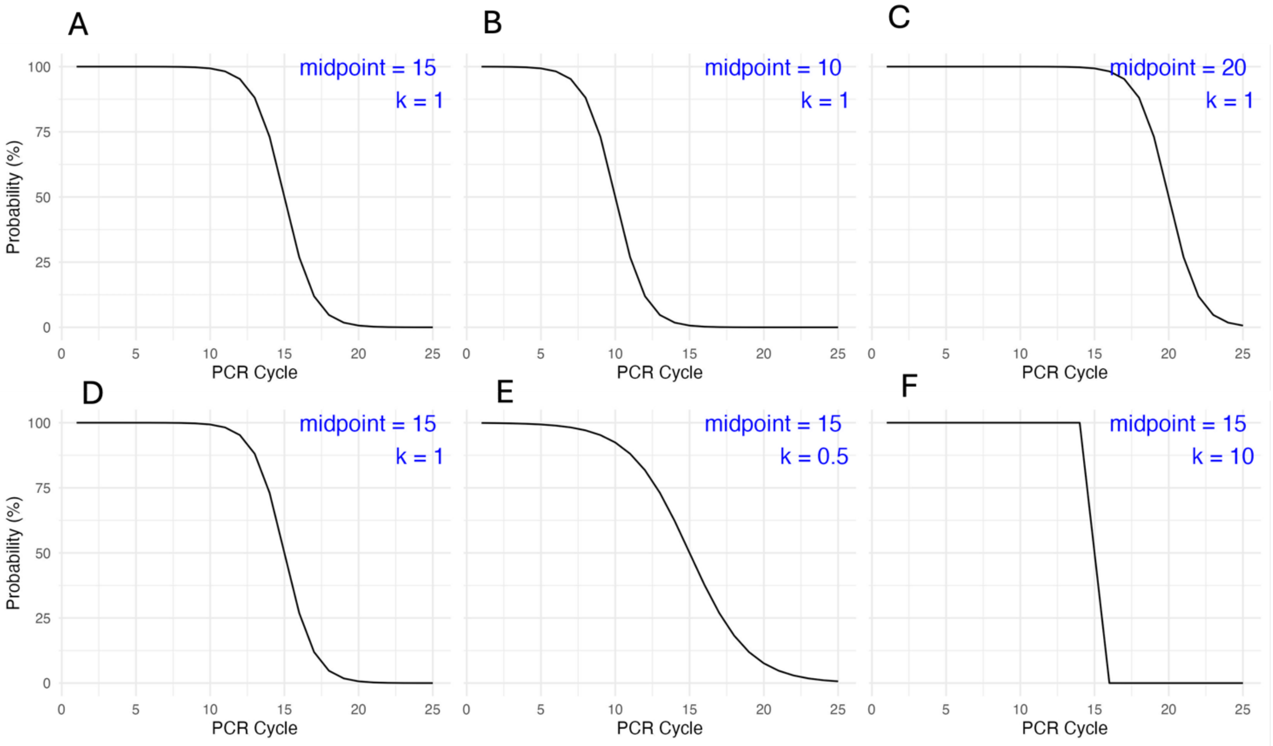


Figure 6. Modelling the efficiency drop via inverted sigmoid function. A) The change of the efficiency, indicated as a probability (Y-axis) of an amplicon to be copied in a PCR cycle, with different midpoint cycles, 15, 20, and 30, respectively. B) The change of the slope shape when different values of k-parameter are used, 0.5, 1, and 10, while the midpoint is constant, 15.

2. *Polymerase error rate - modelling the error-mutations insertions*

Once more Kimura's infinite site model as the one used in the specific mutation rate mode of inserting the mutation during the sample generation is employed to model the polymerase error rate insertion. In the PCR simulation, Poisson distribution determines the number of mutations each newly generated amplicon gets. As the number of copies accumulates during each cycle, the probability of getting an error during amplification grows.

3. *What happens during each cycle?*

The PCR cycling simulation starts with the single fragment entering the first cycle. The fragment is amplified separately through all defined cycles. The fragment is first split into lead and lag units to mimic the real-life PCR amplification where the primer amplification from lead units produces the lag units and vice versa.

Each cycle starts by calculating the amplification efficiency using an inverted sigmoid function, reflecting the dynamic changes in PCR efficiency over time. The efficiency determines, via a binomial distribution, the number of new copies that each of the original units will produce.

The newly formed copies are subjected to a mutation assessment through a Poisson distribution. This step estimates the number of mutations based on the current polymerase error rate and the length of the fragment being amplified for each new copy made. Mutations occur separately on lead and lag unit copies. For a new copy that acquires a mutation, a random position is assigned and recorded. To preserve Kimura's model assumptions, the position can not be mutated twice. Copies that acquired mutations are then considered unique units and are tracked separately to maintain an accurate count of their prevalence.

When new copies of units harbouring mutations are determined, the mutation positions are reverse complemented. This process ensures that mutations in the lead units are accurately mirrored in new copies of the lag units, and vice versa.

By the end of the specified number of cycles, the initial fragment is now a population of unique amplicon units. Each unit is distinguished by specific mutations characterised by its copy number.

The simulation culminates in preparation for the sequencing where a specified fraction of the total PCR products is selected for sequencing. This selection is performed using a binomial distribution that adjusts the copy numbers of each unique unit based on their likelihood of being sampled. This final step simulates the sub-sampling typically seen in laboratory sequencing workflows, where only a portion of the PCR products is analyzed.

Scripts and outputs:

- `PCR.sh` script orchestrates the whole amplicon sequencing PCR process and sequencing prep (described in the chapter Sequencing-prep and Sequencing). The processes described in this chapter are coded in these two C++ programs:
  - `primer_binding_fragments.cpp` identifies the primers' binding positions on each fragment thereby generating the coordinates of all possible amplicons with the specified amplicon length. The process is parallelized by applying the C++ program to a batch of fragments. The output is a csv file with the amplicon coordinates of each fragment and fragment length.
  - `PCR_cpp.cpp` performs PCR cycling simulation. The program operates with the copy numbers of each unique amplicon made (in terms of mutations they acquired during PCR simulation). The main output is a csv table with all unique amplicons generated from all fragments, holding the information on their copy numbers and polymerase error mutation positions.

d. Probe-capture enrichment in WES simulation

In the WES, the fragments harbouring exonic regions are enriched by probes (Figure 6). While enrichment in real life can be quite complex considering the high number of probes and DNA fragments, we simplified the enrichment by using well-known tools. If probes are provided as FASTA sequences, they are first aligned using Bowtie2 (v2.4.4) to the specified human genome, producing BAM files. These BAM files are then converted to BED files with Bedtools (v2.31.1), which detail the probe-binding coordinates on each chromosome.

As the genome fragments were stored in BED files in the previous fragmentation step, the `bedtools intersect` command allows for the screening of overlaps between probe binding coordinates and fragment coordinates. The fragments whose coordinates overlap sufficiently (specified by the `MATCHING_LENGTH`) with the probe coordinates are selected for the short simplified PCR reaction. The simplified PCR should mimic the short indexing PCR essential in the lab preparation for whole exome sequencing. Compared to the simulated PCR in amplicon sequencing, the PCR in WES omits the polymerase error rate insertion but includes the cycle efficiency. However, the efficiency is approximately 100% across the default 8 PCR cycles (advanced settings of the PCR). The high efficiency is achieved by setting the midpoint cycle parameter to be higher than the number of cycles (Figure 9). This ensures that the efficiency remains almost constant during cycling.

The PCR cycling is similar to PCR cycling in amplicon sequencing simulation. The fragment is split into lead and lag units. The current cycle efficiency with the binomial distribution governs the number of new copies produced from lead and lag units. All generated amplicons proceed to the sequencing-prep step and sequencing.

Scripts and outputs:

- `bedtools_parallels.sh` performs the selection of fragments generated in the previous step in parallels. The output is the BED.gz file containing all selected fragments.
- `csv_prep_pcr.sh` converts the BED.gz files into a csv format required for the PCR simulation.
- `PCR_wes.sh` script orchestrates the WES PCR simulation process and sequencing prep (described in the chapter Sequencing-prep and Sequencing). The processes described in this chapter are coded by:
- `PCR_cpp.cpp` - Essentially the script is the simplified version of the script responsible for PCR simulation in amplicon sequencing.

e. Sequencing-prep and Sequencing

The sequencing simulation is executed using the InSilicoSeq tool (v1.6.0) which is adept at generating synthetic reads. Originally developed for shotgun sequencing of full genomes, incorporates a built-in sequencing error model that simulates errors characteristic of various Illumina instruments. A key strength of ISS is its ability to generate realistic read-quality information.

In our simulation, InSilicoSeq has been adapted to produce paired reads specifically from the fragments processed in earlier steps. For each unique fragment, the ISS generates a corresponding read pair: R1 is generated starting from the beginning of the fragment, and R2 is generated in reverse from the end of the fragment. The modification extends to the input files required for read generation including a CSV file specifying the number of read pairs assigned to each fragment, and a FASTA file that contains the sequences of all fragments destined for sequencing. Moreover, the writing of the read files is also modified efficiently splitting the read generation workload between different cores.

The modified Python scripts can be found in the GitHub repository under `modified_iss_script`, where `app_v9.py` replaced the `app.py` script, while `generator_new_reads_writting_v3.py` replaced the `generator.py` script. The rest of the ISS scripts were not modified.

i. Read pair calculation

In the PCR phase of our simulation, each fragment that enters the process is amplified. During amplicon sequencing, the inherent polymerase error rate can lead to the introduction of mutations, producing different variants of the original fragments. Each variant thus has a distinct number of copies, reflecting how often that particular sequence variation was amplified. It's important to note that this specific generation of variants through polymerase errors is typical in amplicon sequencing but does not apply to WES.

For both sequencing methods, the next step involves calculating the relative abundance of each fragment—whether they are variants in amplicon sequencing or simply copies in WES. This relative abundance is determined by the copy number of each fragment, indicating the likelihood of its selection for sequencing.

To allocate the number of read pairs each fragment will receive, we utilize a multinomial distribution. This method assigns read pairs based on the calculated relative abundance of each fragment concerning the total number of reads planned for generation in the simulation.

ii. Optimal length mode - shorter fragments have a greater chance to be sequenced

In our sequencing simulation, we introduced the `OPT_LENGTH_MODE` option to refine the calculation of how many read pairs each fragment receives, incorporating the impact of fragment length. This feature is designed to mimic the real-world sequencing preference where shorter fragments are more frequently sequenced than longer ones.

To model this, we utilize an inverted sigmoid function that adjusts the likelihood of sequencing based on fragment length. The function is calibrated by two key parameters: the midpoint, which defines the fragment length at which the sequencing probability is 50%, and the k-parameter, which controls the steepness of the function, affecting how quickly the probability decreases for longer fragments.

Users can specify the midpoint with several options to suit different experimental needs:

- **`"NO"`**: This option bypasses the length adjustment, relying solely on the relative abundance of fragments for read allocation.
- **`"fixed {integer}"`**: Users can set a specific length as the midpoint, customizing the 50% probability point.
- **`"default"`**: This sets the midpoint to 350 bases.
- **`"median"`**: The median length of all fragments is used as the midpoint

- `"quartile {1|2|3|4}"`: This option allows users to set the midpoint at a specific quartile of the fragment length distribution.

Once each fragment's probability of being sequenced is determined based on its length, this probability is multiplied by its abundance-derived probability. The resulting probabilities are then normalized to ensure they sum to one, maintaining the appropriate distribution across all fragments.

Finally, a multinomial distribution uses these normalized probabilities to allocate the number of read pairs each fragment receives, relative to the total number of reads specified for the sequencing.

Scripts and outputs:

In both `PCR.sh` in amplicon sequencing simulation and `PCR_wes.sh` in WES simulation, the number of read pairs per fragment is calculated by `calcualte_read_pairs.cpp`. The major input is the PCR output file containing the information about the abundance of each fragment. The output is the csv table holding the information of the calculated read-pairs for each fragment.

iii. Generation of the FASTA file containing sequences of all fragments

After the read-pair counts are determined, the sequences of fragments getting at least one read-pair will be generated. The sequences will contain the inserted mutations in the sample generation step, and polymerase-inserted error mutations generated during the amplicon sequencing simulation.

Scripts and outputs:

In both `PCR.sh` in amplicon sequencing simulation and `PCR_wes.sh` in WES simulation, the FASTA file is generated by `process_frags.cpp`. The script is used on batches of fragments in parallel, inserting the initial mutations and PCR error mutations in amplicon sequencing. The output is gz-compressed FASTA.

The generated inputs are then processed by the ISS resulting in the production of paired FASTQ.gz.

VII) Recommendations and warnings

In this chapter, we will cover some general recommendations when it comes to running GENOMICON-Seq. We can only give recommendations as the number of parameters is large, and we were not able to test all possible outcomes.

a. Processing time and memory usage

In our test runs for amplicon sequencing, we worked with a very small HPV16 genome, however, its number of copies was quite large. In WES simulations, we used human chromosome 1 and ran a few tests on the whole human genome. Our advice is to test how different parameters will affect the simulation, although there are a few things to keep in mind.

- **GENOMICON-Seq was made to be run on HPC**. Testing on a smaller number of genome copies in both amplicon sequencing and WES is possible on a personal computer. We tested on macOS Sonome 14.5, 6 CPU Intel core i7 (12 logical cores), 16GB DDR4 and Ubuntu 22.04.3 2.6 12 6 CPU Intel Core i7 (12 logical cores), RAM: 16 GB DDR4, with not more than 500 human chromosome 1 copies, and up to 100 000 HPV genome copies. The numbers do not represent the limit, but only what we have been testing our tool on.
- **GENOMICON-Seq is more CPU-bound than memory-bound**. The greatest memory is required when the number of read pairs is about to be assigned to each fragment. As the multinomial distribution is responsible for this, the greater the number of fragments, the greater RAM is required to accomplish this. In human genomes, the RAM necessary for this step could rise to up to more than 50GB. This step is one of the rare ones that is not parallelized and memory inefficient as multinomial distribution has to be applied to the whole dataset. However, in the future, we are considering other options for the read assignment.
- **The simulation might require around 50GB of working space for the human genome**. However, the final results might occupy around 5GB, again depending on the genome size, its copy number, and the specified number of reads. In amplicon sequencing in general, we are expecting that the genome size used in simulation would be much smaller than human, and up to 5GBs of working space would be necessary. In addtion, SQL_database for the whole human genome takes up to 100GB.
- **Genome size, number of genome copies, mutation number, and, in amplicon sequencing simulation, polymerase error rate affect the processing time the most!** In the final stages of simulation, before the read generation by InSilicoSeq, the FASTA sequences containing all inserted mutations of all fragments that will be sequenced are produced. With the many sequences that would be generated, this process might become a bottleneck.
- In addition to the HPC environment, if available, **NVMe SSD storage enabling a higher number of I/O per second would increase the speed** of the GENOMICON-Seq.
- All simulations we performed, were run on **Educloud** (https://www.uio.no/english/services/it/research/platforms/edu-research/). We used 60 CPUs and 4GB RAM per CPU for amplicon sequencing simulations, and 60 CPUs and 8GB RAM per CPU for WES.
- For WES sequencing, approximately 27 hours were needed to process the complete human genome (hg38), and between 45 minutes and 65 minutes for chromosome 1 only when the number of copies was specified as 2500.
- For amplicon sequencing, with the high number of copies, the high number of mutations, and the high polymerase error rate, on the HPV16

genome, the processing time was between 15 and 30 minutes. However, with larger genomes, the processing time will be different.

b. Parallelization of the processes

The simulations are orchestrated by Snakemake which allows the parallel execution of individual jobs, however, the Snakemake jobs are the simulation steps, where each one is dependent on the output from the previous one. Therefore the Snakemake command should always remain `-j 1`. However, almost every job is parallelized within the rule (each script) splitting the workload of a current job on multiple cores. The parallelization can be achieved when more than 2 samples with the same parameters are about to be run (setting NUM_SAMPLES: 2 or more). Here the number of jobs can be set to 2, however, it will require the precalculation of how many cores should be given to each process (NUM_CORES would be applied to each sample, but the total number of CPUs needed would be NUM_CORES x number of samples). This will require recalculating available memory and CPUs for each parallel process.

c. Known issues and possible bugs

As long as the structure of the input files is as it is described in the input files, the simulation runs should be flawless. However, the simulation has been tested only on Unix-based systems (MacOS and Ubuntu/RedHat).

While the run issue might not occur, please check if all probes or primers have been mapped correctly when they are given as FASTA sequences. Simulation employs Bowtie2 to align probes or primer sequences to the given FASTA file, thereby identifying their binding sites. Alignment is performed with the default settings. If probes or primer sequences do not sufficiently match the reference sequence the alignment setting needs to be adjusted in the Snakefile, allowing more mismatches in the seed, changing the mismatches penalty or switching from end-to-end mode of alignment to local (see the Bowtie2 documentation). Changing these parameters would lead to all probes or primers being mapped correctly.

Another potential issue in amplicon sequencing simulations is generating fewer reads than specified. This can occur especially at very high polymerase error rates. The simulation uses a multinomial distribution to allocate read pairs to each fragment. However, each polymerase error mutation creates a new variant of a fragment with its own copy number. As the error rate increases, the diversity of fragment variants grows, but their individual copy numbers remain low. This high diversity with low copy numbers can complicate the multinomial distribution's ability to accurately allocate read pairs to each fragment variant. Consequently, this may result in a significant drop in the overall number of generated reads compared to the specified number. In our simulations, with a high polymerase error rate, the difference between the number of generated reads and the specified number can be as much as 10%. We are currently working on a better method for read-pair allocation that would me more precise and would not need great computational power.

If any issues occur please report it on the main tool repository on GitHub.