

# **batchMap**

vers 2.0.x

2018-11-27

author: Stephan Fuchs ([fuchss@rki.de](mailto:fuchss@rki.de))

testing: Andre Frühauf ([fruehaufa@rki.de](mailto:fruehaufa@rki.de))

## Content

Content.....	2
1 Short Description.....	3
2 Man Page.....	3
3 Modules and Options .....	6
3.1 General options .....	6
3.1.1 Full workflow - minimal input .....	6
3.1.2 Adapting the workflow using <code>--startm</code> and <code>--endm</code> .....	6
3.1.3 Testing the pipeline using <code>--selftest</code> .....	7
3.1.4 Show version number using <code>--version</code> .....	7
3.2 Input options .....	8
3.2.1 Submit single-end read data using <code>--single</code> .....	8
3.3 Output options .....	8
3.3.1 Select an output directory using <code>--outdir</code> .....	8
3.3.2 Avoiding result file compression using <code>--nogz</code> .....	8
3.4 Module descriptions and options.....	8
3.4.1 Module 0: FASTQ file consistency check .....	8
3.4.2 Module 1: Read trimming.....	9
3.4.3 Module 2: Read mapping .....	10
3.4.4 Module 3: PILEUP .....	10
3.4.5 Module 4: Variant calling .....	12
3.4.6 Module 5: Consensus generation.....	14
4 Output .....	17
5 Usage Example .....	18
6 Known Bugs .....	20
7 Process Flow Chart .....	21
8 License & Disclaimer.....	22
9 References.....	22

## 1 Short Description

*batchMap* integrates algorithms and programs for (i) read trimming, (ii) read mapping, (iii) variant calling, and (iv) consensus generation into a seamless pipeline for bacterial genome reconstruction.

## 2 Man Page

```
batchMap.py [-h] [-t INT] [--startm INT] [--endm INT] [--selftest]
            [--nogz] [--version] [--single] [--outdir STR]
            [--trimpath STR] [--ic STR] [--lead INT] [--trail INT]
            [--sw STR | --mx STR] [--ml INT] [--phred STR] [--bwapath
            STR]
            [--bwasw] [--bwaparam STR] [--stpath STR] [--mpuparam STR]
            [--vspath STR] [--min-coverage INT] [--min-reads2 INT]
            [--min-avg-qual INT] [--min-var-freq FLOAT]
            [--min-freq-for-hom FLOAT] [--p-value FLOAT]
            [--strand-filter INT] [--mapath STR]
            [--cns-max-var-noise FLOAT] [--cns-min-var-freq FLOAT]
            [--cns-report-var-freq FLOAT] [--cns-p-value FLOAT]
            [--report {all,variants,critical}] [--inserts]
            REF_FILE FILE [FILE ...]
```

positional arguments:

REF_FILE	FASTA file containing reference sequence
FILE	input file(s)

optional arguments:

-h, --help	show this help message and exit
------------	---------------------------------

general options:

-t INT	number of cpus (default: 10)
--startm INT	number of module to start your analysis (default: 0):
	0 - fastq file consistency check (input: FASTQ)
	1 - read trimming (input: FASTQ)
	2 - read mapping (input: FASTQ)
	3 - pileup (input: BAM)
	4 - variant calling (input: PILEUP)
	5 - consensus generation (input: VCF 4.1)

```

--endm INT          number of module to end your analysis (default: 5).
                    0 - fastq file consistency check (input: FASTQ)
                    1 - read trimming (input: FASTQ)
                    2 - read mapping (input: FASTQ)
                    3 - pileup (input: BAM)
                    4 - variant calling (input: PILEUP)
                    5 - consensus generation (input: VCF 4.1)

--selftest          unit-like test of essential pipeline functions
--nogz              avoid compression of generated PILEUP and VCF
                    files
--version           show program's version number and exit

input options:
--single            treat FASTQ files as single read-data (default:
                    paired-end read data).

output options:
--outdir STR        use a user-defined output directory (has to be
                    empty or not existent)

TRIMMOMATIC options (TRIMMING MODULE):
--trimpath STR      path to trimmomatic (default:
                    /usr/share/java/trimmomatic-0.32.jar). No spaces
                    allowed in path!
--ic STR            ILLUMINACLIP parameter (default: off)
--lead INT          LEADING parameter (default: off)
--trail INT         TRAILING parameter (default: off)
--sw STR            SLIDINGWINDOW parameter (default: 4:15)
--mx STR            MAXINFO parameter (default: off)
--ml INT            MINLEN parameter (default: 36)
--phred STR         base quality encoding (default: auto-detect)

BWA options (MAPPING MODULE):
--bwapath STR        path to bwa (default: bwa). No spaces allowed in
                    path!
--bwasw             use bwasw instead of bwamem
--bwaparam STR       use additional bwa parameter provided as string
                    (default: off). String has to be enclosed in
                    quotation marks.
--stpath STR         path to samtools (default: samtools). The path has
                    not to contain any spaces!

```

```
--mpuparam STR      use additional samtools mpileup parameter provided
                     as string (default: -B). String has to be enclosed
                     in quotation marks.
```

#### VARSCAN options (VARIANT CALLING MODULE):

```
--vspath STR        path to varscan (default: varscan). The path has
                     not to contain any spaces!
--min-coverage INT   minimum read depth at a position to make a call
                     (default: 10)
--min-reads2 INT     minimum supporting reads at a position to call
                     variants (default: 8)
--min-avg-qual INT   minimum base quality at a position to count the
                     respective read (default: 20)
--min-var-freq FLOAT minimum variant allele frequency threshold to call
                     variants (default: 0.8)
--min-freq-for-hom FLOAT
                     minimum frequency to call homozygote (default: 0.8)
--p-value FLOAT      p-value threshold for calling variants (default:
                     0.01)
--strand-filter INT  ignore variants with greater than 90 percent
                     support on one strand (default: 0). Use 1 for
                     active and 0 for inactive.
```

#### CONSENSUS GENERATION (VARIANT CALLING MODULE):

```
--mapath STR        path to mafft (default: mafft). The path has not to
                     contain any spaces!!
--cns-max-var-noise FLOAT
                     maximal variant allele frequency threshold to use
                     wild type alleles (default: 0.2)
--cns-min-var-freq FLOAT
                     minimum variant allele frequency threshold to use
                     variant alleles. Has to be greater than or equal to
                     value of --min-var-freq (default: same value as
                     --min-var-freq).
--cns-report-var-freq FLOAT
                     minimum variant allele frequency to report in the
                     cns.log file (default: 0.75).
--cns-p-value FLOAT  p-value threshold for calling variants.
                     Has to be smaller than or equal to value of --p-
                     value (default: same value as --p-value).
--report {all,variants,critical}
```

```

report consensus base call decisions (default:
critical).
--cns_inserts INT    consider inserts when consensi are created (default:
1). Set 1 to activate or 0 to deactivate.
--align_inserts INT  consider inserts when consensi are created (default:
0). Set 1 to activate or 0 to deactivate. Different
inserts at the same position are aligned using
mafft. This function is EXPERIMENTAL!

```

### 3 Modules and Options

*batchMap* provides following (optional) tasks for reference-based sequence reconstruction of bacterial genomes:

1. FASTQ file consistency check (own algorithm)
2. quality-based read trimming (*Trimmomatic*<sup>[1]</sup>)
3. read mapping (bwa<sup>[2]</sup>)
4. pileup (samtools mpileup<sup>[3]</sup>)
5. variant calling (varscan<sup>[4]</sup>)
6. consensus generation (own algorithm)

#### 3.1 General options

##### 3.1.1 Full workflow - minimal input

By default, *batchMap* runs all modules from FASTQ file consistency check to consensus generation. For this, the mandatory input consists of one reference file followed by the read data FASTQ file(s). By default, *batchMap* expects paired-end read data divided into two FASTQ files (forward and reverse reads) which may be gz-compressed. To submit multiple FASTQ files, wild cards (\*) can be used. In this case, the file list is alpha-numerically sorted and adjacent pairs of files handled as paired-end read datasets.

Examples:

The minimal mandatory input to run *batchMap*

```
> batchMap.py ref.fasta *.fastq
```

##### 3.1.2 Adapting the workflow using --startm and --endm

The *batchMap* workflow can be adapted by defining a start and/or end module. The module to start (end) with can be selected using the option --startm (--endm) followed by the respective module number (Table 1). Please consider, dependent on the starting module a different input file type is expected.

**Table 1.** Modules and input file types

module number	module	algorithm	input file type	output file type
0	FASTQ file consistency check	own	FASTQ <sup>1</sup>	-
1	read trimming	Trimmomatic	FASTQ <sup>1</sup>	FASTQ
2	read mapping	bwa	FASTQ <sup>1</sup>	BAM
3	pileup	samtools mpileup	BAM	PILEUP
4	variant calling	varscan	PILEUP	VCF
5	consensus generation	own	VCF	FASTA, LOG

<sup>1</sup>GZ compression allowed

Examples:

Running the full workflow on your data (start module: 0 [default]; end module: 5 [default])

```
> batchMap.py ref.fasta *.fastq
```

Applying the pipeline without trimming reads (start module: 2; end module: 5 [default])

```
> batchMap.py --startm 2 ref.fasta *.fastq
```

Applying the pipeline to map data only (start module: 2; end module: 2)

```
> batchMap.py --startm 2 --endm 2 ref.fasta *.fastq
```

### 3.1.3 Testing the pipeline using `--selftest`

The pipeline provides a unit-like test of essential functions by using the option `--selftest`

Running the test, *batchMap* provides a summary of checked functions and test results on screen.

Using this option makes the submission of a reference and FASTQ files obsolete.

Example:

Running the testing routine

```
> batchMap.py --selftest
```

### 3.1.4 Show version number using `--version`

Using the option `--version` the *batchMap* version number is shown on screen. Using this option

makes the submission of a reference and FASTQ files obsolete.

Example:

Accessing the *batchMap* version

```
> batchMap.py --version
```

## 3.2 Input options

### 3.2.1 Submit single-end read data using `--single`

Modules 0, 1, and 2 expect paired-end read data provided in two separate files per dataset (Table 1). To use single-end read data the option `--single` has to be set.

Example:

Analyze single-end read data

```
> batchMap.py --single ref.fasta *.fastq
```

## 3.3 Output options

### 3.3.1 Select an output directory using `--outdir`

By default, *batchMap* creates a subfolder in the *current working directory* according to the pattern *BATCHMAP\_YYYY-MM-DD\_hh-mm-ss*. To specify a user-defined output directory the parameter `--outdir` followed by the designated directory name has to be used. Please note, the named directory has to be empty or not existent.

Example:

Using 'my\_results' folder to store all results generated by *batchMap*

```
> batchMap.py --outdir my_results ref.fasta *.fastq
```

### 3.3.2 Avoiding result file compression using `--nogz`

By default *batchMap* compresses all PILEUP and VCF files generated by module 3 and 4 (Table 1). Compression can be avoided using the option `--nogz`

Example:

Avoid compression of generated PILEUP and VCF files

```
> batchMap.py --nogz ref.fasta *.fastq
```

## 3.4 Module descriptions and options

### 3.4.1 Module 0: FASTQ file consistency check

FASTQ files are used to store read information (Fig. 1). Each read is presented by a quadruplet of lines consisting of

- Line 1 introduced by @; sequence identifier and optional descriptions
- Line 2 raw sequence
- Line 3 introduced by +; optional descriptions
- Line 4 encoded base quality scores.



**Fig. 1.** Representation of two reads in the FASTQ format

### 3.4.2 Module 1: Read trimming

Following options affect the trimming process:

More information on *Trimmomatic* and its options can be found here:

Trimmed datasets are stored in form of FASTQ files (see section 4).

Example:

Trimming paired-end data using MAXINFO and ILLUMINACLIP

```
> batchMap.py --mx 50:0.8 --ic TruSeq3-PE.fa:2:30:10 ref.fasta *.fastq
```

### 3.4.3 Module 2: Read mapping

Module 2 performs reference-based read mapping using *bwa*<sup>[2]</sup>. FASTQ input files containing read data are expected. Paired-end read data (default) have to be provided by two separate files containing the forward and reverse reads, respectively.

Following options affect the mapping process:

- bwapath      Path to *bwa* (default: *bwa*). Please consider, that no spaces are allowed in the in path!
- bwasw        Use *bwasw* algorithm instead of *bwa mem*.
- bwaparam     Use additional *bwa* parameters. The parameter string has to be enclosed by apostrophes (see examples)

More information on *bwa* options can be found at:

<http://bio-bwa.sourceforge.net/bwa.shtml>

Alignment information including mapped and unmapped reads is stored in SAM files which are converted to its binary form (BAM files) using *samtools*<sup>[3]</sup> (for *samtools* options see section 3.4.4). BAM files which are stored (see section 4).

Example:

Mapping reads using *bwasw* and Z-best heuristics

```
> batchMap.py --bwasw --bwaparam '-z' ref.fasta *.fastq
```

### 3.4.4 Module 3: PILEUP

The pileup format is a text-based format summarizing the base calls of reads aligned to a reference sequence. This format facilitates visual display of SNP/indel calling and alignment.

seq1	276	G	22	...T,,,T,...t,,,,...	33;+<<7=7<<7<&<<1;<<6<
seq2	156	A	11	.\$.....+2AG.+2AG.+2AG	<975;:<<<<<<

**Fig 2.** PILEUP representation of a SNP (line 1) and a 2bp insertion (line 2) each supported by three reads.

Each line consists of 5 (or optionally 6) tab-separated columns:

- Column 1      reference sequence identifier
- Column 2      position in sequence (1-based)
- Column 3      reference nucleotide at position
- Column 4      number of aligned reads covering that position
- Column 5      nucleotides of aligned reads at position
- .      match to reference sequence on forward strand

	,	match to reference sequence on reverse strand
	+2AG	2bp insertion of AG between this and next reference position (forward strand)
	+2ag	2bp insertion of AG between this and next reference position (reverse strand)
	-4ACCG	4b deletion of ACCG after this reference position (forward strand)
	-4accg	4b deletion of ACCG after this reference position (reverse strand)
	^	start of read segment followed by ASCII encoded mapping quality
	\$	end of read segment
Column 6		ASCII-encoded base qualities at position (optional)

Module 3 transforms BAM files to PILEUP files using *samtools mpileup*<sup>[3]</sup>.

Following options affect the pileup process:

<code>--stpath</code>	Path to <i>samtools</i> (default: <i>samtools</i> ). Please consider, that no spaces are allowed in the in path!
<code>--mpuparam</code>	Use additional <i>mpileup</i> parameters (default: '-B'). The parameter string has to be enclosed by apostrophes (see examples)

The generated PILEUP files are stored (see section 4).

By default, PILEUP files are compressed to gz archives which can be avoided using the option `--nogz` (see section 3.3.2)

Example:

pileup using a user-defined samtools version (executed using '/bin/my\_samtools/samtools')

```
> batchMap.py --stpath /bin/my_samtools/samtools ref.fasta *.fastq
```

### 3.4.5 Module 4: Variant calling

During variant calling, base call reliability is proven at each position in the read alignment. To store results of the variant calling, the VCF format is frequently used (Fig. 3). VCF file content is introduced by meta-information (starting with ##) and field descriptions (starting with #) followed by data lines which are organized in different columns:

#CHROM	identifier of reference sequence
POS	position in reference sequence (1-based)
ID	semi-colon separated list of unique identifiers
REF	reference base
ALT	allele base (comma separated if different alleles have been detected)
	.                      reference base length < REF    deletion length > REF    insertion
QUAL	phred-encoded quality score for ALT
FILTER	value PASS if all filter where passed. Otherwise filter name explained by meta-information lines (starting with ##FILTER)
INFO	additional information on ALT (separated by semicolons). Tags are explained by meta-information lines (starting with ##INFO) and followed by '=' and a value
FORMAT	used statistic tags (separated by semicolons). Tags are explained by meta-information lines (starting with ##FORMAT)
Sample1	(separated by semicolons) statistics on ALT for Sample1 in the same order as given in FORMAT column

```
##fileformat=VCFv4.1
##source=VarScan2
##INFO=<ID=ADP,Number=1,Type=Integer,Description="Average per-sample depth of bases with Phred score >= 20">
##INFO=<ID=WT,Number=1,Type=Integer,Description="Number of samples called reference (wild-type)">
##INFO=<ID=HET,Number=1,Type=Integer,Description="Number of samples called heterozygous-variant">
##INFO=<ID=HOM,Number=1,Type=Integer,Description="Number of samples called homozygous-variant">
##INFO=<ID=NC,Number=1,Type=Integer,Description="Number of samples not called">
##FILTER=<ID=stl0,Description="Less than 10% or more than 90% of variant supporting reads on one strand">
##FILTER=<ID=indelError,Description="Likely artifact due to indel reads at this position">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality">
##FORMAT=<ID=SDP,Number=1,Type=Integer,Description="Raw Read Depth as reported by SAMtools">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Quality Read Depth of bases with Phred score >= 20">
##FORMAT=<ID=RD,Number=1,Type=Integer,Description="Depth of reference-supporting bases (reads1)">
##FORMAT=<ID=AD,Number=1,Type=Integer,Description="Depth of variant-supporting bases (reads2)">
##FORMAT=<ID=FREQ,Number=1,Type=String,Description="Variant allele frequency">
##FORMAT=<ID=PVAL,Number=1,Type=String,Description="P-value from Fisher's Exact Test">
##FORMAT=<ID=RBQ,Number=1,Type=Integer,Description="Average quality of reference-supporting bases (qual1)">
##FORMAT=<ID=ABQ,Number=1,Type=Integer,Description="Average quality of variant-supporting bases (qual2)">
##FORMAT=<ID=RDF,Number=1,Type=Integer,Description="Depth of reference-supporting bases on forward strand (reads1plus)">
##FORMAT=<ID=RDR,Number=1,Type=Integer,Description="Depth of reference-supporting bases on reverse strand (reads1minus)">
##FORMAT=<ID=ADF,Number=1,Type=Integer,Description="Depth of variant-supporting bases on forward strand (reads2plus)">
##FORMAT=<ID=ADR,Number=1,Type=Integer,Description="Depth of variant-supporting bases on reverse strand (reads2minus)">
#CHROM POS ID REF ALT QUAL FILTER INFO FORMAT Sample1
NC_007795.1 1 . A . . PASS ADP=29;WT=1;HET=0;HOM=0;NC=0 GT:GQ:SDP:DP:RD:AD:FREQ:PVAL:RBQ:ABQ:RDF:RDR:ADF:ADR 0/0:55:29:29:29:0:0%:1E0:93:0:20:9:0:0
NC_007795.1 2 . A . . PASS ADP=29;WT=1;HET=0;HOM=0;NC=0 GT:GQ:SDP:DP:RD:AD:FREQ:PVAL:RBQ:ABQ:RDF:RDR:ADF:ADR 0/0:55:29:29:29:0:0%:1E0:93:0:20:9:0:0
NC_007795.1 3 . A . . PASS ADP=29;WT=1;HET=0;HOM=0;NC=0 GT:GQ:SDP:DP:RD:AD:FREQ:PVAL:RBQ:ABQ:RDF:RDR:ADF:ADR 0/0:55:29:29:29:0:0%:1E0:93:0:19:10:0:0
NC_007795.1 4 . G . . PASS ADP=29;WT=1;HET=0;HOM=0;NC=0 GT:GQ:SDP:DP:RD:AD:FREQ:PVAL:RBQ:ABQ:RDF:RDR:ADF:ADR 0/0:55:29:29:29:0:0%:1E0:93:0:19:10:0:0
NC_007795.1 5 . C . . PASS ADP=28;WT=1;HET=0;HOM=0;NC=0 GT:GQ:SDP:DP:RD:AD:FREQ:PVAL:RBQ:ABQ:RDF:RDR:ADF:ADR 0/0:51:28:28:28:0:0%:1E0:93:0:18:10:0:0
NC_007795.1 6 . G . . PASS ADP=28;WT=1;HET=0;HOM=0;NC=0 GT:GQ:SDP:DP:RD:AD:FREQ:PVAL:RBQ:ABQ:RDF:RDR:ADF:ADR 0/0:51:28:28:28:0:0%:1E0:93:0:17:11:0:0
NC_007795.1 7 . T . . PASS ADP=27;WT=1;HET=0;HOM=0;NC=0 GT:GQ:SDP:DP:RD:AD:FREQ:PVAL:RBQ:ABQ:RDF:RDR:ADF:ADR 0/0:52:27:27:27:0:0%:1E0:93:0:16:11:0:0
NC_007795.1 8 . A . . PASS ADP=27;WT=1;HET=0;HOM=0;NC=0 GT:GQ:SDP:DP:RD:AD:FREQ:PVAL:RBQ:ABQ:RDF:RDR:ADF:ADR 0/0:52:27:27:27:0:0%:1E0:93:0:16:11:0:0
NC_007795.1 9 . C . . PASS ADP=27;WT=1;HET=0;HOM=0;NC=0 GT:GQ:SDP:DP:RD:AD:FREQ:PVAL:RBQ:ABQ:RDF:RDR:ADF:ADR 0/0:52:27:27:27:0:0%:1E0:93:0:15:12:0:0
NC_007795.1 10 . A T . PASS ADP=27;WT=1;HET=0;HOM=0;NC=0 GT:GQ:SDP:DP:RD:AD:FREQ:PVAL:RBQ:ABQ:RDF:RDR:ADF:ADR 0/0:52:27:27:27:0:89,7%:2,347E-10:93:0:15:12:0:0
NC_007795.1 11 . AGG A . PASS ADP=28;WT=1;HET=0;HOM=0;NC=0 GT:GQ:SDP:DP:RD:AD:FREQ:PVAL:RBQ:ABQ:RDF:RDR:ADF:ADR 0/0:51:28:28:28:0:81,3%:3,451E-5:93:0:16:12:0:0
NC_007795.1 12 . A AA . PASS ADP=28;WT=1;HET=0;HOM=0;NC=0 GT:GQ:SDP:DP:RD:AD:FREQ:PVAL:RBQ:ABQ:RDF:RDR:ADF:ADR 0/0:51:28:28:28:0:100%:1E-14:93:0:16:12:0:0
```

**Fig 3.** VCF representation of reference base calls (Positions 1 - 10), SNP (position 10), deletion (position 11), and insertion (position 12). Please consider, that insertions are appended to a base called “anchor base”. Accordingly, deletions are reduced to the preceding called base (“anchor base”). More information about VCF format specifications can be found under: <http://www.internationalgenome.org/wiki/Analysis/Variant%20Call%20Format/vcf-variant-call-format-version-41/>

Module 3 provides variant calling using *VarScan's mpileup2cns*<sup>[4]</sup>. Following options affect the variant calling process:

<code>--vspath</code>	Path to <i>varscan</i> (default: <i>varscan</i> ). Please consider, that no spaces are allowed in the in path!
<code>--min-coverage</code>	Minimum read depth at a position to make a call (default: 10)
<code>--min-reads2</code>	Minimum supporting reads at a position to call variants (default: 8)
<code>--min-avg-qual</code>	Minimum base quality at a position to count a read (default: 20)
<code>--min-var-freq</code>	Minimum variant allele frequency threshold to call variants (default: 0.8)
<code>--min-freq-for-hom</code>	Minimum frequency to call homozygote (default: 0.8)
<code>--p-value</code>	Default p-value threshold for calling variants (default: 0.01)
<code>--strand-filter</code>	Ignore variants with greater than 90 percent support on one strand (default: 0 [inactive])

More information on the *VarScan mpileup2cns* options can be found here:

[http://varscan.sourceforge.net/using-varscan.html#v2.3\\_mpileup2cns](http://varscan.sourceforge.net/using-varscan.html#v2.3_mpileup2cns)

The generated VCF files are stored in the VC subfolder of the result directory. By default, VCF files are compressed which can be avoided by the `--nogz` option (see section 3.1.3).

Example:

Calling variants using a minimal variant allele frequency of 60%

```
> batchMap.py --min-var-freq 0.6 ref.fasta *.fastq
```

### 3.4.6 Module 5: Consensus generation

Module 5 creates a consensus sequence based on VCF file (see section 3.4.5) and reference sequence information. For this, *batchMap* applies its own VCF parser which is optimized for bacterial genomes and provides some advantages such as variant noise filtering, decision logging, and reference-based sequence alignment. However, there are serious disadvantageous, too, which will be discussed below. Please consider that the *batchMap* parser has been only tested with VCF 4.1 format!

Following options affect the consensus generation process:

<code>--mapath</code>	path to <i>mafft</i> (default: <i>mafft</i> ). Please consider, that no spaces are allowed in the path! This option is only of relevance when combined with <code>--align_inserts</code> option.
<code>--cns-min-var-freq</code>	The minimum variant allele frequency threshold to accept a base call (default: same value as <code>--min-var-freq</code> ). Importantly, the given value has to be the same as used for variant calling! This option should be only used if Module 5 is the start module (see 3.1.1).
<code>--cns-max-var-noise</code>	Neglect reference base calls if the variant allele frequency is above this threshold (default: 0.2)
<code>--cns-report-var-freq</code>	Report base call as critical decision (see below), if variant allele frequency is greater than the given value but less than the

	minimum variant allele frequency defined by <code>--cns-min-var-freq</code> (default: 0.75).
<code>--cns-p-value</code>	Maximal p-value to accept a base call (default: same value as <code>--p-value</code> ). Importantly, the given value has to be the same as used for variant calling! This option should be only used if module 5 is the start module (see 3.1.1).
<code>--report</code>	Select report type (default: <code>critical</code> ). If the given value is <code>critical</code> , only positions with critical decisions are reported (see below). If the given value is <code>variants</code> , only positions which are different from the reference and positions with critical decisions are reported. If given value is <code>all</code> , all decisions are reported (larger file size).
<code>--cns_inserts</code>	consider inserts when consensi are reconstructed (default: 1; sequences in consensus files contain inserts). Set 1 to activate or 0 to deactivate.
<code>--aln_inserts</code>	consider inserts when consensi are aligned (default: 0; exclude inserts from aligned sequences). Set 1 to activate or 0 to deactivate. Different inserts at the same position are aligned using <code>mafft</code> . This function is EXPERIMENTAL!

Module 5 creates different files (see also section 4):

<i>alignment.fna</i>	aligned reference (on top) and reconstructed sequences (FASTA format; with gaps)
<i>xxx__cns.fna</i>	consensus sequence of the respective sample xxx (FASTA format; without gaps)
<i>xxx__cns.log</i>	decision report for sample xxx (TXT format). If option <code>--report</code> with value <code>critical</code> is used, this file is only created when critical decisions have been made (see below).
<i>inserts.log</i>	list of inserts across all samples including additional information such as coordinates (TXT format)

### Short introduction to the parser's functionality

The parser checks each position listed in the VCF file regarding the rules listed in Table 2.

**Table 2.** Rules for consensus generation in descending priority (see also Fig. 3)

call <sup>1</sup>	rule	decision	level
R,S,I,D	number of <b>samples not called</b> is 1	rejected	uncritical
R,S,I,D	<b>FILTER</b> is not PASS	rejected	uncritical
R,S,I,D	<b>variant allele frequency</b> and/or <b>p-value</b> are missing (“unreliable”)	rejected	uncritical
R,S,I,D	<b>p-value</b> greater than p-value threshold ( <code>--cns-p-value</code> )	rejected	uncritical
S,I,D	<b>variant allele frequency</b> lower than minimal variant allele frequency threshold ( <code>--cns-min-var-freq</code> ) and lower than reporting threshold ( <code>--cns-report-var-freq</code> )	rejected	uncritical
S,I,D	<b>variant allele frequency</b> lower than minimal variant allele frequency threshold ( <code>--cns-min-var-freq</code> ) but greater than reporting threshold ( <code>--cns-report-var-freq</code> )	rejected	critical
R	<b>variant allele frequency</b> <sup>3</sup> greater than noise threshold ( <code>--cns-max-var-noise</code> )	rejected	critical
R,S,I,D	<b>p-value</b> greater than p-value of another call at same position (“inferior call”) <sup>2</sup>	rejected	critical
?	<b>ALT</b> (call) is not a base call, insert or deletion (“unkown call”)	rejected	critical
R,S,I,D	<b>p-value</b> equal to p-value of another call at same position (“competing call”) <sup>2</sup>	rejected	critical
S,I,D	<b>variant allele frequency</b> greater than reporting threshold ( <code>--cns-report-var-freq</code> )	accepted	critical
R,S,I,D	<b>p-value</b> lower than p-value of another call at same position (“superior call”) <sup>2</sup>	accepted	critical

<sup>1</sup> R = reference base, S = SNP, I = insertion, D = deletion

<sup>2</sup> according to VCF format this should not happen

<sup>3</sup> variant frequencies of gaps and inserts are not considered

All base call decisions (critical and uncritical) are reported to the log file when using the option `--report all` (Table 2). Using the option `--report critical` (default) only critical decisions are reported which reduces the file size. **Critical decisions are introduced by an exclamation mark (!) in the first field and should always be manually revised!**

### Short introduction to the parser’s insert handling

By default, *batchMap* is not considering insertions (only the “anchor bases” are called). If you use the option `--inserts`, inserts are considered. This is not a problem for the consensus sequence representation. However, in the sequence alignment the insert handling can eventually cause problems. That is why this option is indicated as EXPERIMENTAL. If multiple consensus sequences share insertions after the same reference position, it is not problematic. The respective insertions are aligned using MAFFT. However, if insertions of different consensi are not at the same reference position but still overlapping, the alignment should be manually proofed. **To check, if overlapping inserts can infer with the alignment, the *inserts.log* has to be manually inspected!**

### Using an alternative consensus creation program

There are good reasons not to rely on *batchMap*’s VCF parser since (i) it is only tested for VCF 4.1 format, (ii) experienced developers (e.g. the GATK team) is warning to create a own parser, (iii) the VCF format is very complicated. If you don’t want to rely on *batchMap* consensus sequences can be created by other tools such as *vcf-tools*.



## Example

Exclude inserts from consensus sequences

```
> batchMap.py --cns_inserts 0 ref.fasta *.fastq
```

Include inserts in unaligned and aligned consensus sequences (experimental!)

```
> batchMap.py --align_inserts 1 ref.fasta *.fastq
```

Create consensus sequence including indels using VCFtools<sup>[5]</sup> (adapt filenames highlighted in bold)

```
bgzip sample.vcf;  
tabix -f -p vcf sample.vcf;  
cat ref.fasta | vcf-consensus file.vcf.gz > sample_consensus.fasta
```

## 4 Output

If the option `--outdir` (see 3.1.1) is not used, *batchMap* creates a subfolder in the *current working directory* according to the pattern `BATCHMAP_YYYY-MM-DD_hh-mm-ss`. Within this pattern different sub-folders are created to store all relevant data (Table 3). For an example see section 5.

Table 3. The result folder content generated by *batchMap* (shown for paired-end data).

subfolder <sup>1</sup>	file	file content
-	batchmap.log	workflow, parameter, version, input, and result information
-	process.log	standard and error output of integrated programs ( <i>bwa</i> , <i>samtools</i> , <i>varscan</i> )
XX_REF/	reference.fasta <sup>2</sup>	used file containing the reference sequence
XX_TRIMMING/	*_paired.fq.gz <sup>3</sup>	trimmed reads with mates
	*_unpaired.fq.gz <sup>3</sup>	trimmed reads without mates
XX_MAPPING/	*.bam <sup>4</sup>	BAM file containing read alignments
XX_VCALLING/	*.pileup.gz <sup>4</sup>	PILEUP file
	*.vcf.gz <sup>4</sup>	VCF containing variant calling information
XX_CONSENSI/	alignment.fna	aligned reference (on top) and consensus sequences
	*__cns.fna <sup>4</sup>	reconstructed consensus sequence
	*__cns.log <sup>4</sup>	base call decision report
	inserts.log	table of considered insertions in aligned and unaligned form (if <code>--inserts</code> is used)

<sup>1</sup> XX stands for the task number within the selected workflow (see 3.1)

<sup>2</sup> original name of the used reference file

<sup>3</sup> \* stands for the basename of each input file

<sup>4</sup> \* stands for the basename of each dataset (same as input file name in case of single-end data)

## 5 Usage Example

Using *batchMap* (user input is highlighted in bold):

```
> batchMap.py --inserts ref.fasta *.fq.gz
preparing output folder ...
  BATCHMAP_2017-06-29_13-03-50
processing FASTQ dataset 1 of 2 ...
  sample1_R1.fastq.gz
  sample1_R2.fastq.gz
  checking ...
    564270 paired-end reads
processing FASTQ dataset 2 of 2 ...
  sample2_R1.fastq.gz
  sample2_R2.fastq.gz
  checking ...
    537372 paired-end reads
processing reference ...
  ref.fasta
  checking ...
  indexing ...
trimming datasets ... [2/2]
read mapping ... [2/2]
pileup ... [2/2]
variant calling ... [2/2]
generating consensi ... [2/2]
writing consensi ...
aligning consensi ...
writing alignment ...
logging ...
cleaning and compressing ...
```

Following folders and files are created:

```

❏ BATCHMAP_2017-06-29_13-03-50
  ❏ 01_REF
    ❏ ref.fasta
  ❏ 02_TRIMMING
    ❏ sample1_R_1_paired.fq.gz
    ❏ sample1_R_1_unpaired.fq.gz
    ❏ sample1_R_2_paired.fq.gz
    ❏ sample1_R_2_unpaired.fq.gz
    ❏ sample2_R_1_paired.fq.gz
    ❏ sample2_R_1_unpaired.fq.gz
    ❏ sample2_R_2_paired.fq.gz
    ❏ sample2_R_2_unpaired.fq.gz
  ❏ 03_MAPPING
    ❏ sample1_R_.bam
    ❏ sample2_R_.bam
  ❏ 04_VCALLING
    ❏ sample1_R_.pileup.gz
    ❏ sample1_R_.vcf.gz
    ❏ sample2_R_.pileup.gz
    ❏ sample2_R_.vcf.gz
  ❏ 05_CONSENSI
    ❏ alignment.fna
    ❏ inserts.log
    ❏ sample1_R__cns.fna
    ❏ sample2_R__cns.log
    ❏ sample2_R__cns.fna
    ❏ sample2_R__cns.log
  ❏ batchmap.log
  ❏ process.log

```

Please consider

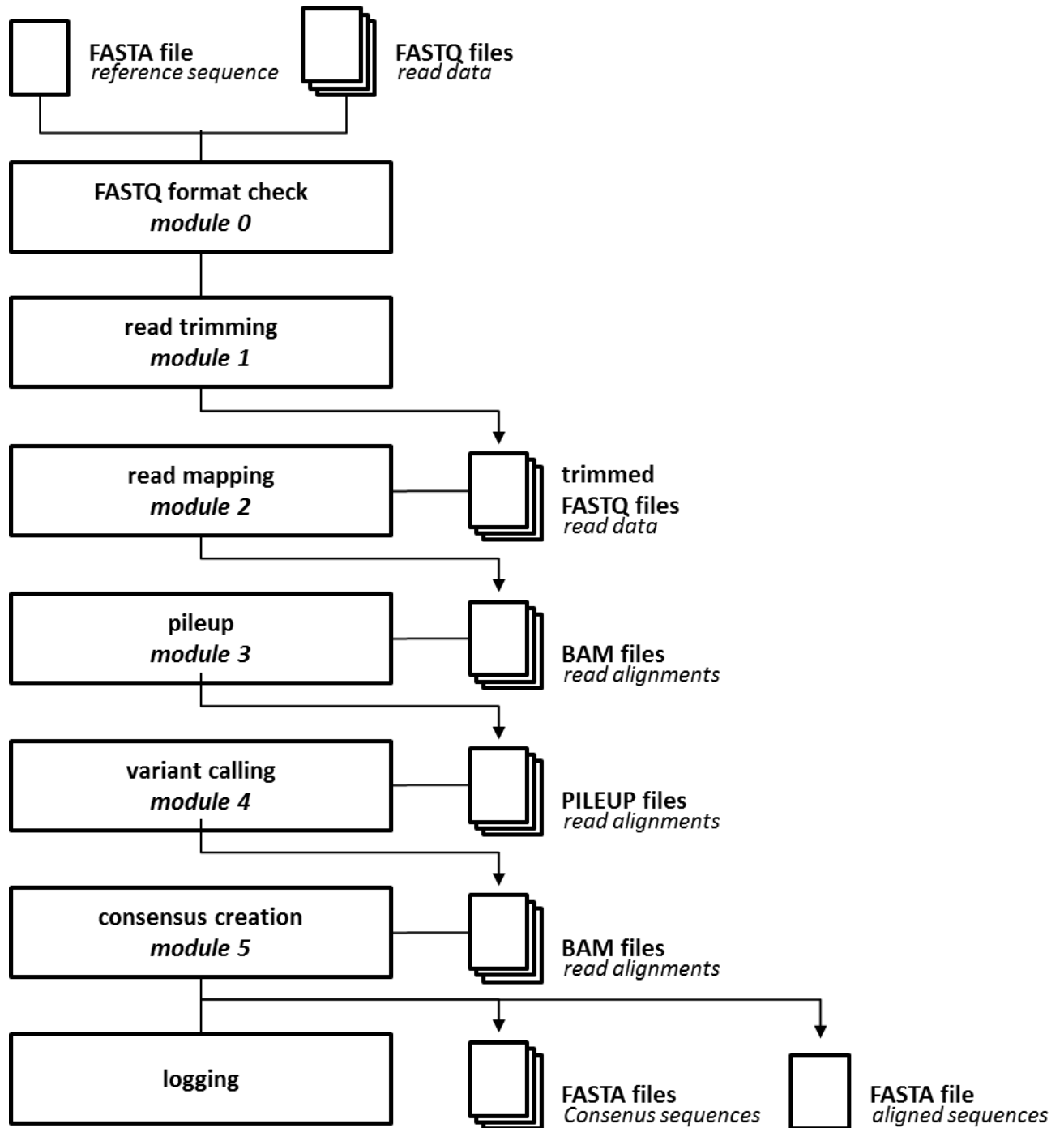
- the *inserts.log* file is only created when using the experimental option `--aln_inserts 1` or `--cns_inserts 1` (see section 3.4.6)
- the *xx\_\_cns.log* files (xx stands for sample file name) are generated only if something is to report
- **red files contain important information which should be considered by the user after running the pipeline!**

## 6 Known Bugs

Known bugs will be solved soon.

If too less reads are mapping to the reference (e.g. after mapping reads obtained from KPC to a *S. aureus* genome), the PILEUP file will be empty which crashes the subsequent tasks.

## 7 Process Flow Chart



## 8 License & Disclaimer

Copyright © 2017 Stephan Fuchs, RKI (FG13), fuchss@rki.de

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

## 9 References

- 1 Bolger, A. M., Lohse, M. & Usadel, B. Trimmomatic: a flexible trimmer for Illumina sequence data. *Bioinformatics* **30**, 2114-2120, doi:10.1093/bioinformatics/btu170 (2014).
- 2 Li, H. & Durbin, R. Fast and accurate long-read alignment with Burrows-Wheeler transform. *Bioinformatics* **26**, 589-595, doi:10.1093/bioinformatics/btp698 (2010).
- 3 Li, H. A statistical framework for SNP calling, mutation discovery, association mapping and population genetical parameter estimation from sequencing data. *Bioinformatics* **27**, 2987-2993, doi:10.1093/bioinformatics/btr509 (2011).
- 4 Koboldt, D. C. *et al.* VarScan 2: somatic mutation and copy number alteration discovery in cancer by exome sequencing. *Genome Res* **22**, 568-576, doi:10.1101/gr.129684.111 (2012).
- 5 Danecek, P. *et al.* The variant call format and VCFtools. *Bioinformatics* **27**, 2156-2158, doi:10.1093/bioinformatics/btr330 (2011).