SHORE Manual

Version 0.5.0beta

Stephan Ossowski
Korbinian Schneeberger
Felix Ott
Jörg Hagmann
Sebastian Bender
Alf Scotland

@ Max Planck Institute for Developmental Biology
Tübingen, Germany
01.03.2010

Contents

SHORE is a mapping and analysis pipeline for short DNA sequences produced on Illumina Genome Analyzer and Hiseq 2000 or Life Technology SOLiD platforms. It is designed for projects whose analysis strategy involves mapping of reads to a reference sequence. This reference sequence does not necessarily have to be from the same species, since weighted and gapped alignments allow for accuracy even in diverged regions.

SHORE's mapping strategy is best-hit-only, a conservative approach, though there is no limit in the number of best hits. Paired end and mate pair information is used to increase mapping quality. Additionally SHORE provides error models for Illumina GA characteristics, e.g. GC coverage bias and error models for alignment issues in e.g. repeats or low complexity sequence.

SHORE does not only predict SNPs. It provides various prediction algorithms for genomic polymorphisms, i.e. SNPs, structural variants (indels, CNVs, unsequenced regions), SNPs and SV prediction in heterozygous or pooled samples, as well as peak detection for ChIP-Seq analysis and quantitative analysis of mRNA-Seq and sRNA-Seq.
Future updates will incorporate QPALMA for spliced alignments and additional models for quantitative Analysis of RNA-Seq (e.g. detection of differentially expressed sRNA and mRNAloci) and ChIP-Seq and mapping and analysis of BS-seq reads. For a detailed list of future developments please check the roadmap on the last page of this manual.

SHORE stores read data, mapping alignments and result files in a pre-defined folder structure, which makes it possible to keep track of all intermediate steps and, if desired, repeat parts of the analysis. The predefined folder structure has advantages and disadvantages. While there is no freedom in data structuring when applying SHORE, it makes handling of large projects comprising multiple flowcells more convenient. It is in the nature of such projects that it can take weeks to gather all information, while the initial mappings have to be performed as soon as the first flowcell is finished. This requires an extendable mapping and analysis approach keeping all information in a transparent structure. Of course, also smaller projects can benefit from such a data partitioning.

SHORE was designed to run on a multi core server (32 or 64 bit) with Linux or Mac-OS (10.5). Required memory depends on the application and the reference sequence. Medium sized genomes (e.g. D. melanogaster, A. thaliana) can be analyzed with 2-8GB RAM, large genomes (e.g. H. sapiens) with 8-24GB RAM. SHORE is multi-threaded to take advantage of multi-core architectures. SHORE implements several alignment tools (e.g. GenomeMapper, bwa, bowtie, eland) each coming with their own hardware requirements.

Document overview:
Installation instructions can be found in Section 1 and a Quick Start Guide in Section 2. Sections 3.1 and 3.2 provide an overview of analysis strategy and data structure. Section 4 provides an in-depth help for SHORE's subprograms.

Disclaimer:
SHORE is currently under beta testing and please be aware of the fact that inappropriate usage or input files will lead to unpredictable results. We would, of course, be most grateful for feedback about errors and usability of SHORE. The software license for SHORE will be announced with the first release.

1 Prerequisites

1.1 Download and Installation

Download the latest SHORE release from

`https://sourceforge.net/projects/shore/` or `http://1001genomes.org/downloads/`

or check out the git repository to compile SHORE from the very latest source:

```
> git clone git://shore.git.sourceforge.net/gitroot/shore/shore
```

1.2 Installing SHORE using the binary file

Navigate to the folder where you stored the downloaded file:

```
> tar xzvf shore_<platform>_<version>.tar.gz
```

SHORE is now installed.

1.3 Installing SHORE using the source files

Navigate to the folder where you stored the downloaded file

```
> tar xzvf shore_source_<version>.tar.gz
> cd shore_source_<version>
```

(or navigate to your shore git folder).

See the file `INSTALL` in the SHORE directory for further installation instructions.

1.4 Download and install a mapping tool

SHORE can currently be run with five different mapping tools. The native tool of SHORE is GenomeMapper, which can be downloaded from:

http://1001genomes.org/downloads/

Navigate to the folder where you stored the downloaded file:

```
> tar xzvf genomemapper.tar.gz
> cd genomemapper
> make
```

GenomeMapper is now installed. Note Mac OS users might have to change the name of the Mac OS Makefile. See the GenomeMapper manual for further details and details on parallelization of GenomeMapper.

In addition to GenomeMapper, SHORE supports Eland (included in GA Pipeline), BWA (http://bio-bwa.sourceforge.net/), Bowtie (http://bowtie-bio.sourceforge.net/index.shtml) and Novocraft (http://www.novocraft.com). All tools are available as free download for academic use. These tools have to be installed according to their manuals. Note that we do not guarantee that future developments will be available for all of these tools. The selection of the mapping tool depends on the type of project as each tool has some pros and cons. This list is based on our everyday's struggle experiences and might be different to yours:

GenomeMapper:
Pro: Adjustable for high number of mismatches and gaps, long reads, accurate alignments (Needleman-Wunsch like)
Con: Gapped alignments are slower than ungapped alignments.
Recommended for: genome re-sequencing, high polymorphism density, spliced alignments and any other applications

Version: 0.4.2 or higher *BWA:*
Pro: Fast and small memory requirement, unlimited mismatches and gaps.
Con: Currently no spliced alignment or BS-seq support
Recommended for: any Illumina GA or SOLiD data (except see above)

Version: 0.5.7 or higher *Bowtie:*
Pro: Small index size due to Burrows-Wheeler compression
Con: Only up to three mismatches and no gaps allowed
Recommended for: large genomes and low polymorphism density, servers with small memory

Version: 0.12.3 or higher *Eland:*
Pro: Its fast.
Con: Only up to two mismatches and no gaps allowed
Recommended for: small RNA mapping, rapid testing

Version: Illumina GA Pipeline 1.3 or 1.4 *Novocraft:*
Pro: Unlimited number of mismatches and gaps.
Con: Seems to be slow, especially for long reads and high number of mismatches/gaps.
Recommended for: currently not recommended due to insufficient testing of the SHORE wrapper for novo

Version: 2.05.31 or higher

1.5 Installation of R

SHORE makes use of R to provide visualization of run statistics. In order to use this feature, R has to be installed and the installation path has to be added to the $PATH environment variable.
Note that it is not necessary to have R installed to run SHORE.

1.6 Environment variables

Since SHORE supports external mapping tools, it needs to know where these are located on the actual system. At least one environment variable describing the folder where the mapping tool is stored has to be set to run SHORE. Here we describe this exemplarily for the bash and show how to set up SHORE to run with GenomeMapper. Though all four mapping tools could be added at the same time. For tcsh or any other shell, users have to set the environment variables in an equivalent way. Add to your local .bashrc file ($HOME/.bashrc):

required:

```
export GENOMEMAPPER=/PATH/TO/GENOMEMAPPER/INSTALLATION/FOLDER/
```

optional:

```
export BWA=/PATH/TO/BWA/INSTALLATION/FOLDER/
export BOWTIE=/PATH/TO/BOWTIE/INSTALLATION/FOLDER/
export ELAND=/PATH/TO/ELAND/INSTALLATION/FOLDER/
export NOVO=/PATH/TO/NOVOCRAFT/INSTALLATION/FOLDER/
```

Alternatively the mapping tools may be added to your PATH variable:

```
export PATH=$PATH:'/PATH/TO/GENOMEMAPPER/INSTALLATION/FOLDER/'
```

1.7 System requirements

Minimum:
Linux or Mac-OS (10.5 or later)
Intel Xeon/Core Duo/Core i7 or AMD Athlon/Opteron
2GB RAM (only for small to medium sized genomes)
4TB Hard-disc

Recommended:
Linux or Mac-OS (10.5 or later)
2 x Quad Core Intel Xeon or AMD Opteron
16-128GB RAM
12-256TB Hard-disc

1.8 TMP directory

Some files need to be sorted before SHORE can use them. Before using these input files SHORE checks if they are sorted in the appropriate way and in case of inconsistencies SHORE will start to sort them. E.g., initially alignments are sorted by read ids, but to analyze them they need to be sorted by the alignment location. Therefore sorting is

required. SHORE implements its own sorting program. By default this program uses `/tmp` if the files to be sorted are large (which is usually the case). Therefore, it can happen that the disc space in `/tmp` is not sufficient. We recommend at least 200GB. This can be achieved by setting the TMPDIR environment variable to a file system with sufficient space.

```
export TMPDIR=/PATH/TO/A/FOLDER/
```

2 Quick Guide

2.1 A quick example

2.1.1 Preprocess the reference

```
> shore preprocess
  -f ~/downloads/phiX.reference_sequence.fa
  -i ~/shore_index/phiX_seed12/
  -W -b -C -B
```

-f defines the reference sequence
-i describes the folder hosting the shore IndexFolder
-W additionally create bwa index
-b additionally create bowtie index
-C additionally create colorspace index (available for bwa and bowtie)
-B additionally create index for BS-seq alignments

2.1.2 Convert Illumina GA-Pipeline Bustard folder to SHORE flat file format

```
> shore import -v Bustard -e Shore
-a genomic -i 1001 -b my/bustard/folder
-o ~/phiX_resequencing_project/run_01/ -D
```

`-v` defines the input format, `-e` the output format, `-a` starts a genomic re-sequencing project, `-i` sets a flowcell id, `-b` bustard folder produced by GA-Pipeline, `-o` output folder, `-D` disable Illumina's chastity filter

2.1.3 Convert Illumina FastQ files to SHORE flat file format

```
> shore import -v Fastq -e Shore
-a genomic -i 1001 -x sequence_1_1.fq  -y sequence_1_2.fq
-o ~/phiX_resequencing_project/run_01/ -Q illumina
```

`-v` defines the input format, `-e` the output format, `-a` starts a genomic re-sequencing project, `-i` sets a flowcell id, `-x` fastq file of the first read, `-y` fastq file of the second read, `-o` output folder, `-Q` format of the quality string.

2.1.4 Convert SOLiD csfasta and QV files to SHORE flat file format

```
> shore import -v Solid -e Shore
-a genomic -i 1001 -F reads_F3  -R reads_R3
-o ~/phiX_resequencing_project/run_01/
```

-v defines the input format, -e the output format, -a starts a genomic re-sequencing project, -i sets a flowcell id, -F prefix of the F3 csfasta file, -R prefix of the R3 csfasta file, -o output folder.

2.1.5 Map all reads of a Illumina flowcell folder

```
> shore mapflowcell -f ~/phiX_resequencing_project/run_01/
  -i ~/shore_index/phiX_seed5/phiX.reference_sequence.fa.shore
  -n 4 -g 3 -c 8 -p
```

-n max edit distance, -g max gaps, -c number of CPUs, -p indicate paired end data, -f flowcell folder, -i *.shore file in the IndexFolder.

2.1.6 Map all reads of a SOLiD flowcell folder

```
> shore mapflowcell -f ~/phiX_resequencing_project/run_01/
  -i ~/shore_index/phiX_seed5/phiX.reference_sequence.fa.shore
  -v bwa -C
  -n 4 -g 3 -c 8 -p
```

-n max edit distance, -g max gaps, -c number of CPUs, -p indicate paired end data, -v name of the mapper, -C indicate that reads are in colorspace, -f flowcell folder, -i *.shore file in the IndexFolder.

2.1.7 Correct for paired-end information

```
> shore correct4pe -l ~/phiX_resequencing_project/run_01/7 -x 250 -e 1
```

-l lane, -x max insert size, -e library number, -s indicate colorspace data, -m indicate Illumina Mate-Pair libraries.

2.1.8 Merge all read alignments of a project

```
> shore merge -p ~/phiX_resequencing_project/run_01
  -d ~/phiX_resequencing_project/AlignmentFolder/
```

-p list flowcell folders, -d output folder.

2.1.9 Consensus analysis

```
> shore consensus -n phiX -r
  -f ~/shore_index/phiX_seed5/phiX.reference_sequence.fa.shore
  -o ~/phiX_resequencing_project/AlignmentFolder/Analysis_01
```

```
-i ~/phiX_resequencing_project/AlignmentFolder/map.list
-a /shore_install_folder/scoring_matrix_hom.txt -b 0.51
```

`-n` sample name, `-r` plot statistics using R, `-f` *.shore file in the index folder, `-o` output folder, `-i` map.list file, `-a` scoring matrix, `-b` minimum agreement of base calls (here for homozygous samples).

2.1.10 Check the result files

Run statistics plots can be found in

`~/phiX_resequencing_project/AlignmentFolder/Analysis_01/ConsensusStatistics/`

Analysis result files can be found in

`~/phiX_resequencing_project/AlignmentFolder/Analysis_01/ConsensusAnalysis/`

Have a look at 'quality_variants.txt' and 'quality_reference.txt'. The result files and formats will be explained in more detail in a later chapter. If you have R installed and included in your $PATH variable, you can find two PDFs in

`~/phiX_resequencing_project/analysis/analysis_01/ConsensusStatistics/`

One shows the GC content dependent coverage, the other one plots read errors by base quality in various plots. Find more details on run statistics plots in a later section.

3 SHORE overview

3.1 What is SHORE and what is SHORE for?

SHORE is a data analysis and management application for short DNA/RNA reads produced by the Illumina Genome Analyzer or Life Technology SOLiD platforms. It is developed at the Max Planck Institute for Developmental Biology, Tübingen, Germany.
SHORE is designed to support different sequencing applications including genomic re-sequencing, ChIP-Seq, mRNA-Seq, sRNA-Seq and BS-seq. The SHORE pipeline comprises all necessary steps for sequence analysis including quality filtering of raw reads, adapter clipping for sRNA-seq, mapping of reads against a reference genome, repeat analysis, correction for read pair information, quantitative analysis of ChIP or transcriptome data.
Several file format converters ensure the compatibility of SHORE with other widely used file formats like fastq, qual, sam, bed and gff. SHORE was developed for applications in Arabidopsis thaliana but has been successfully used with other genomes, including human, D. melanogaster, C. elegans, maize and several bacterial genomes.

3.2 Overview of SHORE's data structure

SHORE has a predefined structure of folders and files. One folder, referred to as IndexFolder, stores all information about the reference sequence including the indices (created and used by mapping tools), GC content and low complexity sequence analysis. A second folder, referred to as ProjectFolder, contains all information about a certain sequencing project.
The IndexFolder is kept separate from the ProjectFolder, because one IndexFolder can be used for multiple ProjectFolders, e.g., different re-sequencing projects (ProjectFolders) can refer to the same reference sequence (IndexFolder).
The IndexFolder is populated using `shore preprocess`. It will store a fasta file named <original fasta file>.shore. This file is used as prefix to find all needed index files for SHORE. It is not necessary to keep the original fasta file.
The ProjectFolder has a pre-defined folder structure. All SHORE programs require this structure, though this should not be too confusing, as all folders are created automatically by SHORE.
All files in the ProjectFolder are ASCII formated files and can be viewed, moved and manipulated. For example it is possible to use SHORE just for read mapping and then convert the SHORE alignments into a different file format (e.g. SAM for SNP detection with SAMtools).

3.2.1 The ProjectFolder and the read data

Folder structure of the ProjectFolder for read storage:

```
ProjectFolder/FlowcellFolder/LaneFolder/ReadFolder/LengthFolder
```

ProjectFolder/	This is the home folder for a sequencing project.
FlowcellFolder/	A project starts with the sequencing of one or more flowcells. Each flowcell is stored in a separate folder in the ProjectFolder, the so-called FlowcellFolders.
LaneFolder/	Each flowcell consists of one to eight lanes, each of them represented as a single folder, named 1-8. An additional folder called bad_quality will be created containing all reads which did not pass the quality filter.
ReadFolder/	The read folders separate the read pairs, in case of paired-end or mate-pair data. In folder "1" are the reads from the first sequencing run and in "2" the reads from the second, respectively. In "single" are reads from the first and from the second run which lost their read partner in the quality filtering process. In a single-end run, all reads passing the quality filter will be stored in "single".
LengthFolder/	SHORE can trim reads before mapping according to their base quality at the read ends and therefore reads can be of different lengths. Each length has its own sub-folder which name is equal to the length, prefixed by "length_".

For example, a folder could be named like:

```
phiX/EAS67_0018_FC12398AAXX/5/1/length_80
```

3.2.2 The ProjectFolder and the alignment data

After the reads are mapped, the resulting alignments (mappings) can be merged into the so-called AlignmentFolder. This folder stores all information and results of the alignment analysis:

```
ProjectFolder/AlignmentFolder/AnalysisFolder/ResultFolder
```

ProjectFolder/	This is the home folder for a sequencing project.
AlignmentFolder/	The name of the folder is user specified and will host the merged results of the read mapping which is used as input for `shore consensus`, `shore coverage` and `shore peakup`. Merging the alignments will introduce redundancy. The alignments are stored twice, it is recommended to delete the merged file after is has been analyzed.
AnalysisFolder/	Will be created by `shore consensus`, `shore coverage` and `shore peak`. Several AnalysisFolders can be created to store data from different analysis runs.
ResultFolder/	AnalysisFolders have two different subfolders, the ResultFolders. In case of 'shore consensus' there are two ResultFolders called 'ConsensusAnalysis' which stores the alignment analysis results (e.g. SNP calls, peaks ...) and 'ConsensusStatistics' whichstores general project statistics, like read quality.

It is possible to store the AlignmentFolder or the AnalysisFolders in a different location than the ProjectFolder.

3.3 SHORE's file formats

Any output generated by SHORE will usually be written to various text files that contain a number of tab-delimited columns. Typing `shore fmt` will display a quick reference on SHORE's file formats.

3.3.1 Read file format

Read files can be found in the LengthFolders in the ProjectFolder. They are called 'reads_0.fl'. They will be created by `shore import`.
The tab delimited entries are:

<id>	ID is build up of the run id (4 characters), lane (1 character), tile (3 characters), x value within the image (5 characters), y value (5 characters). In contrast to the GAPipeline the numbers are concatenated and the run id is added at the beginning. The run id makes reads unique between flowcells within one project. Run ids must not start with zero.
<sequence>	DNA sequence
<pe>	Flag, '1' or '2', first read or second read of a read pair. '0' for a single read.
<Illumina quality values>	Quality values described in a later section
<Sanger quality values>	Sanger calibrated quality values described in a later section
<Chastity values>	Illumina chastity values defined as Intensity(max)/(Int(max) + Int(second))

3.3.2 Alignment file format

SHORE alignment files are typically called `map.list`, `map.list.1` or `map.list.2`. They can be found in the LengthFolders or, when applying paired-ends, in the ReadFolders after correcting for paired-end information (`shore correct4pe`).
The tab delimited entries are:

<chr id>	Each chromosome has an internal id, they are simply numbered after their occurrence within the reference sequence file starting from 1. Translation to the native chromosome name can be found in the *.shore.trans file in the IndexFolder.
<pos>	Left-most position of the mapping relative to the reference sequence.

<alignment>	Matches are reported as a single character, mismatches are represented by two characters surrounded by brackets. The first character represents the reference base, the second character the sequenced base. Deletions are represented as '-'.
<read id>	See 'Read file format' section.
<strand>	'D' for forward and 'P' for reverse hits (direct and palindromic, respectively).
<mismatches>	The number of mismatches in the alignment.
<hits>	The total number of genomic positions the read is aligned to.
<read length>	Length of the read.
<offset>	Number of bases at the beginning of a read that have not been aligned.
<pe flag>	'1' or '2' for paired-reads, '0' for singletons. Other values describe the state of paired reads after correcting for pe information.
<Illumina quality values>	Quality values are described in a later section.
<Sanger quality values>	Sanger calibrated quality values described in a later section
<Chastity values>	Illumina chastity values defined as the highest intensity divided by the sum of the highest and the second highest intensity of a single base.

3.3.3 Consensus result file formats

Analyses of resequencing projects are performed by `shore consensus`. Multiple result files are produced for SNPs, indels, CNVs, reference like bases and other types of predictions. All of them can be found in the ConsensusAnalysis folder. Currently 'SHORE consensus' provides three types of consensus predictions: 1. Decision tree based approach as described in Ossowski et al. Genome Research 2009 and 2. Empirical scoring matrix approach to be described in a future publications. 3. SNP prediction on transcriptomic or other quantitative data to be described in a future publication. The output files of the second approach have 'quality_" as name suffix. The third approach produces all files from the first two approaches but uses a slightly different scoring scheme. The following list gives an overview of all column types which can be found in the result file. Note that not all columns apply to each of the predictions:

<name>	Project name or name of sequenced sample
<chr>	Chromosome identifier
<pos>	Position within the chromosome
<start>	First position of a prediction within the chromosome (for predictions longer than just one base pair, e.g. indels)
<end>	Last position of a prediction
<length>	Length
<ref base>	Reference base
<cons base>	Consensus base (i.e. SNP call). Heterozygous SNPs and SNPs from pooled samples are divided into 'major allele' and 'minor allele'
<seq>	Sequence (i.e. inserted or deleted sequence)

<quality>	Quality of a predicted feature (ranging from 0 to 40)
<read type>	part of the reads that were used for prediction: repetitive/nonrepetitive and core/complete
<support>	Number of reads supporting a predicted feature. For heterozygous SNPs and SNPs from pooled samples the sequenced bases are divided into 'major support' and 'minor support'
<concordance>	Ratio of reads supporting a predicted feature to total coverage (excluding quality masked bases). Heterozygous SNPs and SNPs from pooled samples are divided into 'major concordance' and 'minor concordance'
<max qual>	Highest base quality supporting a prediction.
<avg hits>	Average number of alignments of all reads covering this genomic position. (see section 'Repeat analysis based on short read alignment' for details)
<repeat count>	Number of repetitive positions in the range of the prediction (i.e. long deletion)
<ambi count>	Number of ambiguous positions in the range of the prediction ('N').
<exp cov>	Expected coverage at a locus defined by repeat analysis and GC content. (Average expected coverage, if the prediction describes a range rather than a single location.)
<obs cov>	Observed coverage at a locus as defined by the read alignment. (Average expected coverage, if the prediction describes a range rather than a single location.)
<obs/exp ratio>	Ratio of observed to expected coverage is used to identify CNVs or highly over-sampled regions (often seen for rDNA clusters) in the genome.
<obs/exp ratio max>	The maximum of <obs/exp ratio>.
<GC>	GC content of a feature, averaged for ranged features.
<GC max>	Maximal GC content within the given range.
<cvp count>	Copy variable positions (variation between instances of repeats) are an indication of duplications or CNVs.

The following listing describes the different prediction files and their columns. This description only reviews some of the major aspects to help to understand the files. It is important to note that all of them are predictions. Especially CNVs and duplications are only indicating abnormalities from the observed mapping data compared to what would be expected under ideal sequencing circumstances. Moreover, CNV and duplication predictions have so far been implemented for single end data only, and should be carefully validated.
More information on the prediction algorithms can be found in Ossowski et al. Genome Research 2008.

Homozygous SNP calls (decision tree approach) (`homozygous_snp.txt`)
All positions with a base call different to the reference. Base calls require a concor-

dance of >= 80% and a support of at least three non-repetitive reads. Due to statistical sampling and sequencing biases, the accuracy of these call is affected by heterozygous SNPs as well when sequencing heterozygous samples.

<name> <chr> <pos> <ref base> <cons base> <read type> <support> <concordance> <max qual> <avg hits>

Homozygous small indels calls (decision tree approach) (`deletions.txt , insertions.txt`)
Deletions (length depends on the number of gaps allowed in the mapping process) called from the alignments. Parameters are identical to those from t he homozygous SNP predictions.

<name> <chr> <start> <end> <length> <seq> <read type> <support> <concordance> <avg hits>

Heterozygous SNP and small indel calls (decision tree approach) (`heterozygous_call.txt`)
All positions with at least 25% of the bases different to the majority call. This file includes minor alleles of indels.

<name> <chr> <pos> <ref base> <major allele> <major support> <major concordance> <minor allele> <minor support> <minor concordance> <unique prb> <avg hits>

SNP and small indel calls in pooled samples (decision tree approach) (`minor_allele_call.txt`)
All positions with either a homozygous SNP/Indel or a variant with a minor allele frequency of >= 2% are stored in this file. Due to the sequencing error of approximately 1% this can result in a high number of false positives. The minimum minor allele frequency should be adjusted according to the number of individuals in the sample. As a rule of thumb it should be greater than (100 / num samples / 2), i.e >= 10% for 5 pooled samples. Note that in case of homozygous variants the minor allele is 'X' meaning no minor allele found. Positions with homozygous reference calls are not stored at all to save space.

<name> <chr> <pos> <ref base> <major allele> <major support> <major concordance> <minor allele> <minor support> <minor concordance> <avg hits>

Homozygous reference calls (decision tree approach) (`reference.txt`)
Reference like positions called from the alignments. Parameters are identical to those from the homozygous SNP prediction.

<name> <chr> <pos> <ref base> <cons base> <support type> <support> <concordance> <max qual> <unique prb> <avg hits>

Copy Variable Positions, CVPs (`copy_variable_position.txt`)

A duplication in the sequenced sample will map to the same locus in the reference sequence as the origin where it was generated from. If there is a (slight) difference between the original position and the duplication, there will be positions which look like het calls. These positions are called CVPs. Positions (so-called CVPs) with two different bases due to mislocated alignments of repetitive sequences are an indication of duplications. CVPs are classified according to their expected repetitiveness. If a position is expected to be unique but contains different bases this can indicate a duplication of a former unique region.

<name> <chr> <start> <end> <length> <cvp count> <obs cov> <exp cov>

CNV (`CNV.txt`)
Copy number variation is predicted based on two major criterias. Requires strong skew between observed and expected coverage of at least 40bp of length and the existence of at least one CVP within this interval.

<name> <chr> <start> <end> <length> <cvp count> <obs cov> <exp cov>

Duplication (`duplication.txt`)
Duplications are predicted similar to CNVs described above, however the reference sequence has to be mostly unique within the duplicated interval and the length has to be greater than 250bp. Thus duplication predictions are more reliable than CNV predictions.

<name> <chr> <start> <end> <length> <cvp count> <obs cov> <exp cov>

Unsequenced regions (`unsequenced.txt, supplementary_data/unseq_cn.txt, supplementary_data/uns`
(`supplementary\_data/unseq\_core.txt`)
Unsequenced regions are called, if a region of one or more bp is continuously uncovered by reads. However this does not necessarily mean that this is a deletion. It can indicate long deletions, insertions, (highly) polymorphic regions or a bias in the sequencing coverage. To account for biases, namely the GC content influencing coverage, we report the average and maximum GC content and the expected coverage within the unsequenced interval. In addition the number of 'N' in the interval is reported to indicate if an unsequenced region is solely due to unreliable reference sequence. Other predictions (SNPs, small indels) in the vicinity of unsequenced regions are unreliable due to an increased probability of alignment issues. In addition we provide two files showing regions with absence of non-repetitive reads and absence of 'core reads'.

<name> <chr> <start> <end> <length> <ambi count> <GC> <GC max> <repeat count> <exp cov>

Oversampled regions (`supplementary_data/oversampled.txt`)
Some highly repetitive genomic regions like rDNA or centromeric repeats are not annotated correcly in the reference sequence. Often the repeat is only represented by one

copy. This leads to unexpectedly high covergae in this regions, indicating that several copies of the repeat mapped to the same reference sequence. The total amount of repeat instances can be estimated using the observed vs. expected coverage ratio. Other predictions (SNPs, small indels) in the vicinity of oversampled regions are unreliable due to an increased probability of erroneous alignments.

<name> <chr> <start> <end> <length> <ambi count> <GC> <GC max> <repeat count> <obs cov> <exp cov> <obs/exp cov ratio avg> <obs/exp cov ratio max>

SNPs and short indels predictions (scoring matrix approach)
(`quality_variant.txt`)
We have developed an empirical scoring matrix with 12 features describing e.g. the quality of reads, the quality of alignments and the likeliness of wrong calls due to other features in the vicinity of a position. These features are used to calculate a quality value for each position/call of the genome ranging from 0 to 40. The scoring matrices can be adjusted by the user, however this is not recommended. There are predefined matrices for different prediction types (e.g. homozygous, heterozygous or pooled samples).

<name> <chr> <position> <ref base> <cons base> <quality> <support> <concordance> <avg_hits>

Reference positions predictions (scoring matrix approach)
(`quality_reference.txt`)
Same as above, just for reference like positions instead of SNPs and indels.

<name> <chr> <position> <ref base> <cons base> <quality> <support> <concordance> <avg_hits>

3.3.4 Statistics

Running `shore consensus` with option -r creates a ConsensusStatistics folder. It contains the files run_statistics.pdf and GC_by_Cov_Statistics (if R was installed correctly). These figures are useful to check for inconsistencies in the analysis.

'Mismatches per Read' describes the fraction of reads which could be mapped without mismatches, with 1 mismatch, 2 mismatches and so on. Indels are counted as mismatches, an indel of length 2 will be counted as 2 mismatches. The black numbers above the orange spots are the absolute read counts, whereas the y-axis describe the percentage of reads. Mismatches can be created by sequence divergence and sequencing errors.

'Number of hits in genome per read'. This plot indicates the repetiveness of the reference sequence based on the number of alignments. It plots how often a certain number of alignments per read was observed. In this example the repeat has approx. 40 instances in the reference sequence.

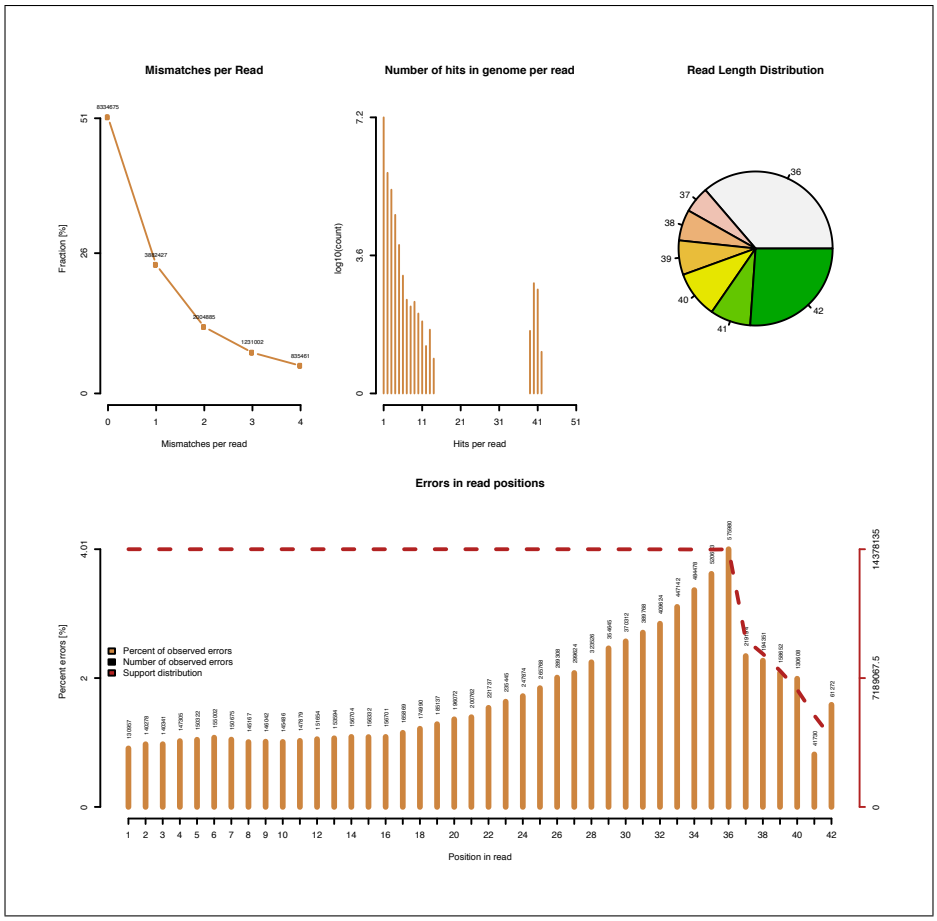

Figure 1: First page of run_statistics.pdf featuring 4 different plots.

'Read Length Distribution'. If read trimming was switched on in `shore illumina2flat`, there will be reads of different length. In Figure 1 read trimming was performed on reads of length 42 down to a minimal length of 36.

'Errors in read positions' plots the percentage of sequencing errors per read position. In the example of Figure 1 the number of read errors decays to the read end. This might be surprising, but is a result of the read trimming. Read trimming clips the last bases if these are of bad quality. Therefore if the last bases are not trimmed they are of at least moderate quality and less error prone. The red line indicates the support (how often a certain read position was counted), also a result of read trimming.
Errors are assessed in uniquely mapped regions, where consensus calling is possible. Read bases which contradict with the consensus are reported as errors.

'Errors in mapped reads at uniquely mapped positions, excluding ambiguous calls - Quality' plots the Sanger scaled Illumina quality value (of a called base) versus the observed percentage of errors. You could refer to this as the probability for an error at a given quality value. Absolute numbers of observations are plotted as dashed red line, while they are written numerical above the orange bars. Note that it depends on the choice of quality (adjusted with `shore illumina2flat`) whether this

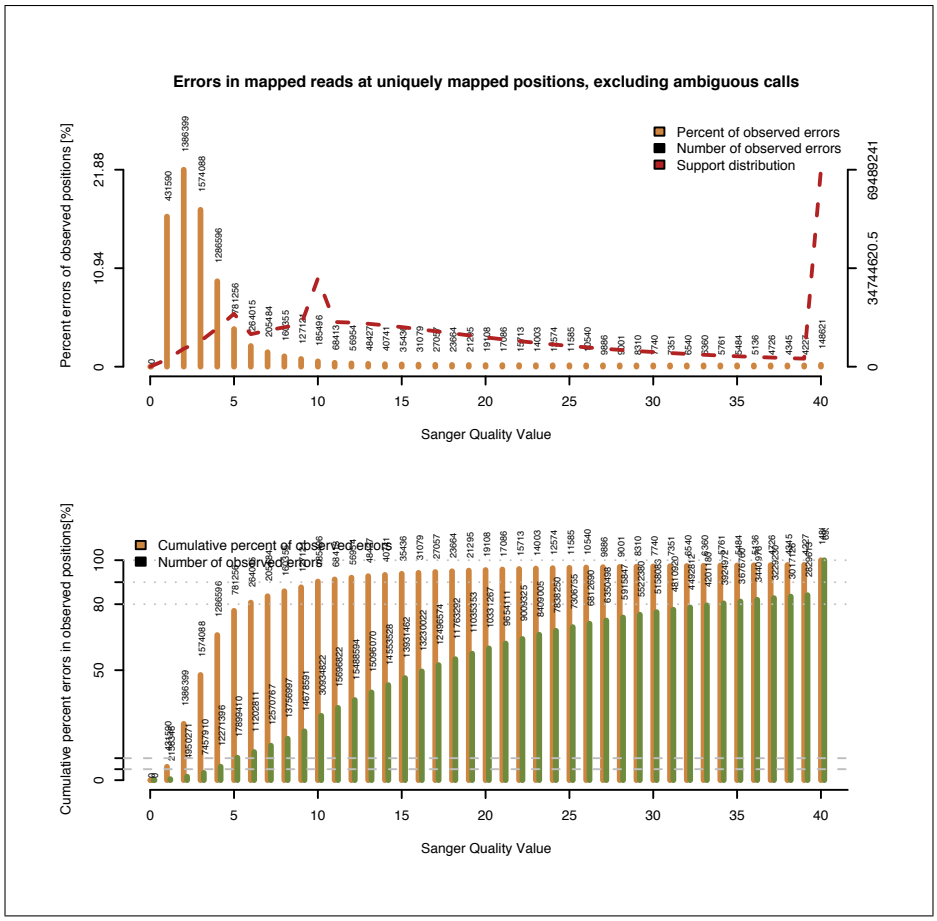

Figure 2: Second page of run_statistics.pdf.

plot shows qraw (prb) or qcal. If the choice was qcal you will not see any values below 0. It is somewhat surprising that in the example of the first plot in Figure 2 base calls with a quality value below 0 seem to perform better than those with a quality value of 0. Though we have observed this pattern quite often.

The second plot on the second page shall help to adjust the quality cutoff for masking. Masking is a very effective way to reduce noise which affects variant/consensus calling. Though we have found that there is nothing like the optimal cutoff value for all runs, since the variance between Illumina data can be large. Therefore the cutoff for base masking by quality value might be re-adjusted.

The plot can be read like this: check the green and orange bar at (let's say) quality value (x-axis) equals 4. There you can see that the green bar is just above the second gray dashed line which indicates the 10% mark. This means setting the quality cutoff to 4 would have masked around 10% of correct bases. On the other hand, nearly 80% of the wrong bases would have been masked as well. This is indicated by the orange bar at quality value of 4.

A much stricter quality cutoff would have been 32, which would have removed 80% of the correct data while it also would have removed 99% of the error prone bases.

Obviously it is desirable to find a quality value (x-axis value) where the green bar is preferably small and the orange bar preferably large. It is a trade-off between strict

masking (remove most of the errors) and weak masking (keeping the correct values). This decision depends on the application.
Note, the erroneous and correct bases have been assessed on unique, well covered and reference like stretches of the sample.

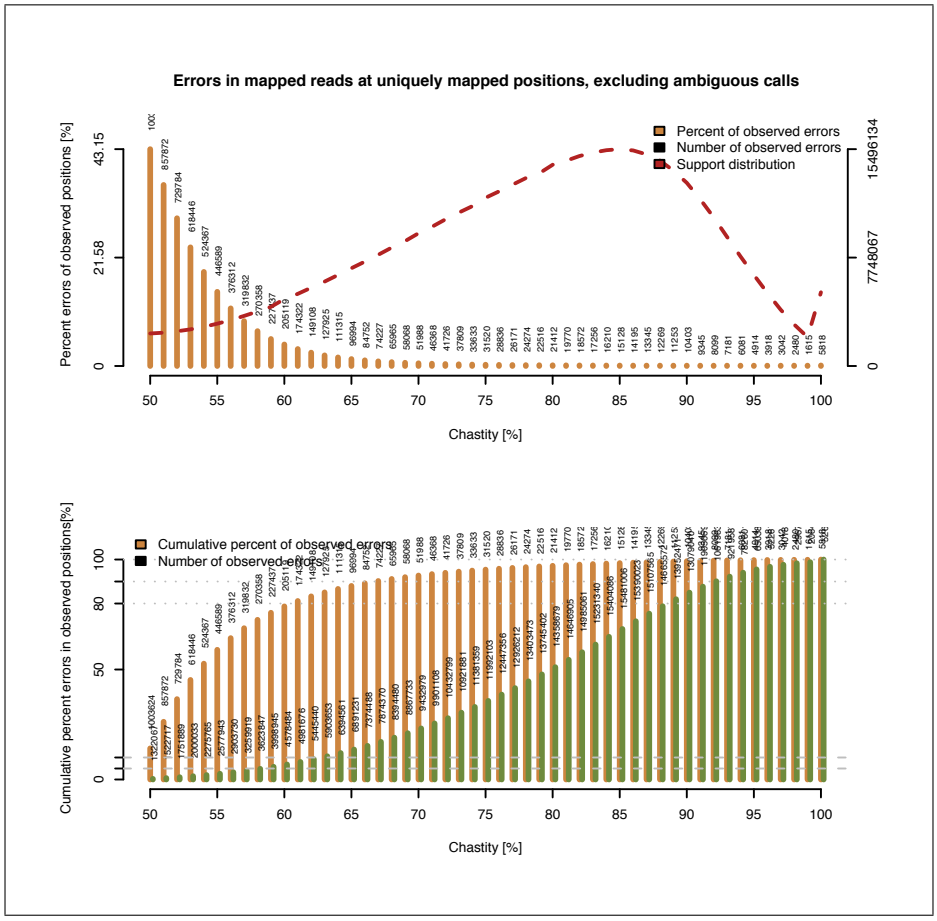

Figure 3: Third page of run_statistics.pdf.

'Errors in mapped reads at uniquely mapped positions, excluding ambiguous calls - Chastity' The third page shows the same kind of plots as the second page, using this time the Chastity score as a base quality measure. The Chastity value was introduced by Illumina's GAPipeline and is used to differentiate between clusters which are interfered by other clusters and those which are not. It is defined as the highest base intensity of a sequenced bases devided by the sum of the highest plus the second highest intensity of the specifc base. Therefore the Chastity score can reach values from 0.5 to 1, or in percent from 50 to 100.
We find this value a useful support for the quality values. Again you can adjust the Chastity cutoff the same way as for the other two values.
Note, all these plots combine read 1 and read 2. It would be desirable to have them separately, and this will be implemented in one of the next releases of SHORE.

'Coverage and GC content at uniquely mapped sites'. After running `shore consensus` there will be a coverage analysis of various GC contents, found in GC_by_-

Cov_Statistics.pdf. The plot prints the average coverage (height of an orange bar) by local GC content (x-axis value). The local GC content is defined as the number of Gs and Cs divided by the number of As, Cs, Gs and Ts within a window which size is adjusted in `shore preprocess`.
Ideally the average coverage is equal at all different GC contents, such a plot would have orange bars of the same height. This is not the case and it has been shown that average coverage scales with the GC content (for regions of low GC content) and has a reverse correlation for regions with high GC content. It is unclear where this comes from, and there has been Illumina data which is not influenced by such a bias at all, though correcting for the bias is a step which potentially can help in variant calling and is essential in calculating the expected coverage value. This value is essential for the dectection of CNVs.
The red dashed line shows the support (occurrence) of a certain GC content window in the unique portion of the genome. Average coverage should be similar to the sequence depth.

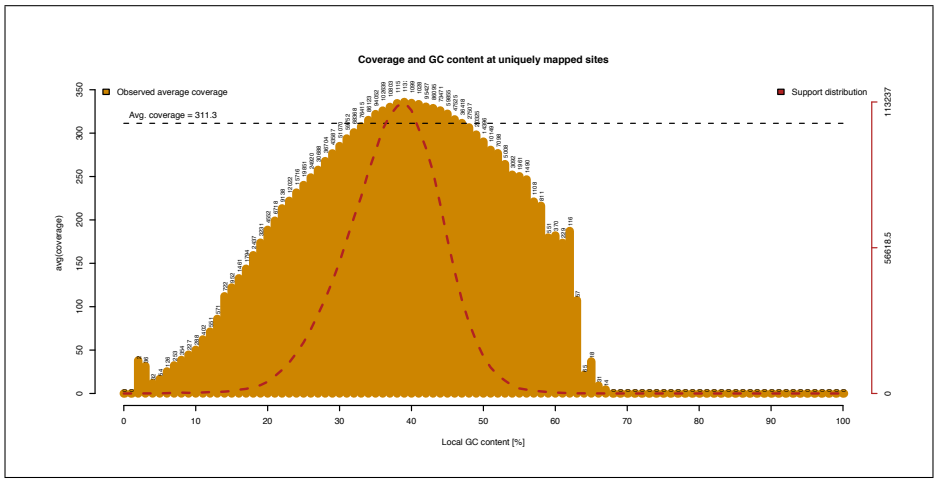

Figure 4: First page of GC_by_Cov_Statistics.pdf.

3.3.5 Quality values

SHORE is using quality values. There are two different types of quality values generated by the GAPipeline. An additional scaling for both of them has been introduced by the Sanger Institute. We integrated both quality values and the Sanger scaling into SHORE.

The GAPipeline reports the 'qraw' (or 'prb') value and the 'qcal' value (from .qseq files) as per base qualities. There are 4 qraw (prb) values and one qcal value per base. It is the user's decision which quality value SHORE shall use (default and highly recommended: qcal). SHORE will automatiaclly scale the Illumina qualities into Sanger phred format and store both Illumina and Sanger scaled values in the reads file.
SHORE incorporates an additional third quality value, the Chastity score. It is defined in the GAPipeline and used to filter for overlapping clusters on the flowcell identified by double intensity peaks. SHORE uses Chastity more general as a measure of ambiguity

of a called base.
All three quality values are represented in both read files and alignment files. All three values are stored as ASCII characters, but the dynamic range is different. The decimal values of the characters of the different strings can be interpreted like this:

<Illumina quality values>
Depending on whether qraw or qcal was chosen, the range is from

	real	ascii
qraw	-5 - 40	59-104
qcal	0 - 40	64-104

Offset is 64. Subtracting of 64 will give the actual quality value, this offset is necessary to move the values in a region of displayable characters in ASCII format. GAPipeline version 1.3 or higher does not use negative qualities anymore (qraw ranges from 0 to 40).

<Sanger quality values>
The colleagues from the Sanger Institute introduced a log formula to linearize the quality value such that the highest Solexa quality value and the 'Sanger' quality score are asymptotically identical. The range is from

	real	ascii
qraw	0 - 40	33 - 73
qcal	0 - 40	33 - 73

Offset is 33.

<Chastity values>
The range is from

	real	ascii
chastity	50-100	40-90

Offset is -10. We are aware of the fact that the different scalings are redundant, the Sanger's scaling can always be inferred from Illumina's scaling, and (only with new GAPipeline version 1.3 or higher) vice versa. Future releases of SHORE will calculate this on the fly to reduce storage space.

4 Running SHORE

4.1 Overview

Each of the different sub-programs of SHORE can be started by typing shore and adding the command name of the sub-program. Starting SHORE without a subprogram name will list all sub-programs:

```
> shore

> shore CMD
```

Executing `shore CMD` will then give you the program specific command line options, explained in the next section.

4.2 SHORE subprograms

Each of SHORE's subprograms work on a specified level within the folder hierarchy. For example `shore mapflowcell` works on the level of the FlowcellFolder, whereas `shore merge`, a program to merge mapping results of a whole project, works on the level of the ProjectFolder.

4.2.1 shore preprocess

`shore preprocess` creates the mapping indices, calculates local GC content and repetitiveness. In addition, SHORE will create a new copy of the fasta file of the reference sequence featuring adjusted chromosome/contig ids and output all files in the IndexFolder. The repeat analysis is time intense. But it is only necessary to run it once for each reference sequence. The repeat analysis is resource and time intensive because it compares each possible n-mer of the reference sequence against the whole genome. Therefore SHORE can be used without a preprocessed repeat analysis and instead rely on the repeat analysis done on the fly while mapping "real" reads.

Usage:

Mandatory:

`-f` `--fastafile=<arg[,...]>`
Fasta file(s) containing all reference sequences

`-i` `--indexfolder=<arg>`
IndexFolder. Output folder for all SHORE relevant files.

Mapping indices:

`-s` `--seed=<arg[,...]>` (Default: '12')
Seedlength(s) for mapping indices (5-13). GenomeMapper and Vmatch support seeds from 5 to 13 bp, Novocraft seeds of 11 to 15. bwa, bowtie and eland do not use this parameter. Seed length is a trade-off between sensitivity and runtime.

`-b` `--bowtie`
Activate Bowtie support

`-n` `--novo`
Activate Novocraft novoalign support

`-e` `--eland`
Activate Eland support

`-v` `--vmatch`
Activate Vmatch support

`-W` `--bwa`
Activate BWA support

`--bwa-construction-algorithm=<arg>` (Default: 'is')
BWA construction algorithm. Possible options are: is (up to 2GB databases), bwtsw (databases larger than 10MB). For further details see 'bwa index'.

`-g` `--no-genomemapper`
Inactivates GenomeMapper

BS-Seq (bisulfite treated DNA):

`-B` `--bsseq`
Turns on indexing for BS-seq experiments. Only genomemapper and novo are supported. BS-seq indices are calculated in addition to the normal indices for genomemapper and novo.

Sequence complexity:

`-c` `--complexity=<arg>` (Default: '9')
Window size in bp for sequence complexity analysis. Sequence complexity is an important factor for alignment quality. Low complexity sequence and most importantly homopolymers often lead to wrong alignments and thus wrong base calls. Sequence complexity is therefore determined for each position taking into account a window ranging 9bp (default) upstream and downstream.

`-w` `--gccontent=<arg>` (Default: '101')
Odd window size in bp for GC content measuring. Apart from repeats we found that local GC content influences read coverage. Calculating the differences between expected read count and observed read count as a measure of copy number variation detection has to consider sequence biases.

SOLiD support:

`-C` `--build-colorspace-index`
Build color-space index

Other Options:

`--maxsize=<arg>`
Split into multiple indeces if the reference sequences exceed this size in megabytes. Multiple indeces can be passed to `mapflowcell` to perform a stepwise mapping. Mainly useful for mapping versus reference sequences larger than 4GB, which are not supported by most mapping tools. Note that other tools like `consensus` may not support more than a single index.

`--headers`
Include the complete fasta headers in the *.trans file (default is first word)

4.2.2 shore import

This program converts Illumina GAPipeline folder structure, Illumina fastq files or SOLiD csfasta files into SHORE read file format. SHORE requires a special read file format typically called `reads_0.fl` or flat-file. The program will accept input as generated by the GAPipeline programs *goat* or *bustard*, `FASTQ` files and `CSFASTA` files.

`shore import` will create the necessary files and folder structures. It has to be started for each flowcell separately (selection of lanes, length and read type is possible). In order to have `shore import` work properly for Illumina reads, the GAPipeline must have been run with default parameters, concerning the output format of quality values.

Input formats of the importer are specified using option -v. Available importers are:

- *Bustard*:
  Input generated by the GAPipeline (bustard/goat) or SCS programs.
- *Fastq*:
  FastQ files. Some users prefer Illuminafastq files as standard output from the GAPipeline.
  The header lines (those startting with @) of the fastq files need some attention. They have to be in a specific format. It is the format you get when you run GERALD (or GOAT) from the GAPipeline with the SEQUENCE_FORMAT option set to `--fastq`:

  `@<Flowcell id>:<lane>:<tile>:<image x-value>:<image y-value>/<read>`

  Note that it is not important what is written as flowcell id nor will the importer look at the read tag. The read id will be built up on lane, tile, and the x and y values.
- *Solid*:
  SOLiD F3 and R3 csfasta and (optionally) QV files
- *Shore*:
  SHORE reads_0.fl files. This importer can be used to re-filter or trim reads which are already in SHORE format

Usage:

Mandatory:

`-v` `--importer=<arg>` (Default: 'Bustard')
Importers: Bustard, Fastq, Shore, Solid

`-e` `--exporter=<arg>` (Default: 'Shore')
Exporters: Shore, Console. 'Shore' automatically creates SHORE's folder structure and files. 'Console' will write all converted reads to standard output.

`-a` `--application=<arg>`
The type of sequencing project. Supported applications: genomic, mRNA, ChIPseq, sRNA.

`-i` `--run-id=<arg>`
Set a run ID. This can be picked between 1000 and 9999. This ID will be added to all read IDs from this flowcell, otherwise the read IDs can collide between flowcells. Use a run ID only once per project.

Bustard importer:

`-b` `--bustard-folder=<arg>`
Bustard folder, *_qseq.txt files The complete path of the Bustard folder that should be translated into SHORE read format. This folder should at least contain the `*_qseq.txt` files. Other files (`*_sig2.txt`, `*_prb.txt`) may be used if available.

`-l` `--lanes=<arg[,...]>` (Default: '1,2,3,4,5,6,7,8')
Allows for sub-selection of lanes, e.g. to only extract lane 1, 2, 4 and 7 type -l 1,2,4,7.

`-D` `--disable-illumina-filter`
Unfiltered reads - do not use the GAPipeline filter. By default, only reads that pass the 'chastity filter' defined by illumina will be extracted. If this option is activated, any read will be used regardless of it's quality. As an alternative, the `-c` switch may be used to specify SHORE's custom chastity filter.

`-r` `--use-qraw`
Use qraw values instead of qcal (requires prb files). To use this option, *bustard* or *goat* must have been run with the `--with-prb` switch. Not recommended.

Fastq importer:

`-Q` `--quality-type=<arg>` (Default: 'illumina')
Quality type provided in fastq files (either "sanger" or "illumina")

`-x` `--fastq-files=<arg[,...]>`
List of fastq files for the first run (read 1)

`-y` `--fastq-files-paired-end=<arg[,...]>`
List of fastq files for the second run (read 2 of paired-end runs) [NOTE: Same file order as in the -x option required]

Shore importer:

`--input=<arg[,...]>`
Input reads_0.fl files or SHORE style flowcell folders

Solid importer:

`-F` `--F3prefix=<arg>`
Prefix of F3 csfasta and _QV.qual file. Both files have to be stored in the same folder. QV file is optional.

`-R` `--R3prefix=<arg>`
Prefix of R3 csfasta and _QV.qual file. Both files have to be stored in the same folder. QV file is optional.

Shore exporter:

`-o` `--flowcell-folder=<arg>`
Output (Flowcell) folder, will be created. Hosts all data of a flowcell. This folder must not yet exist in the ProjectFolder and will be created.

`-B` `--batch-size=<arg>`
Divides the length folders into batches that contain <batch-size> reads. Useful for parallelization of alignment step and faster sorting of read.

`-z` `--read-compression`
Compress read files. Using this option, SHORE will generate compressed `reads_0.fl.gz` files instead of the plain text `reads_0.fl` files.
`--no-filtered-compression`
Do not compress trash files. By default, all reads that go into the `bad_quality` folder will be compressed in the gzip format.

Read filtering:

`-n` `--max-Ns=<arg>` (Default: '4')
Maximum number of ambiguous base calls per read. Reads that contain more than the given number of 'N's will be filtered out.

`-g` `--no-lowcomplexity`
Turn off low complexity filter. By default, all homopolymer reads are filtered. Reads are considered to be homopolymers if more than 90% of the bases are equal.

`-c` `--chastity-filter`
Use custom chastity filter (implies '-D'). Chastity filtering is used to remove reads that result from overlapping clusters. The default GAPipeline filter is equivalent to using `import` with parameters `-c -C 25:1 -V 0.6`.

`-C` `--chastity-filters=<arg[,...]>` (Default: '12:2,25:5')
Filter setup for chastity filtering, e.g. "12:2,25:5" means: allow up to two chastity violations in the first 12 bases, OR up to five in the first 25. What is considered a violation may be specified through `-V`.

`-V` `--chastity-violation=<arg>` (Default: '57')
Threshold for chastity violations (percentage)

Read trimming:

`-m` `--max-length=<arg[,...]>`
Maximum read length(s). Setting this below the cycle number of the run will cause SHORE to truncate every read to the given length before further processing. Different values can be specified for read 1 and read 2.

`-k` `--minimal-length=<arg[,...]>`
Minimal read length - switches on read trimming. An algorithm similar to a poly-A trimming algorithm detects the range of low quality bases at the end of the reads and trims them to the minimal length of k. Different values can be specified for read 1 and read 2.

`-q` `--quality-cutoff=<arg>` (Default: '5')
Quality cutoff for read trimming
`--destructive-trim`
Truncate ends of trimmed read. If not selected reads are not truncated but moved to the length folder of the trimmed length.

Adapter clipping (application = sRNA):

`-d` `--adapter-sequence=<arg>`
Adapter sequence (please specify first 12 bp). Specifies the first 12 bp of the 3' adapter of the sRNA library. The first 8 bp will be used to detect the beginning of the adapter to cut out the sRNA. The other 4 bases are used only if the first 8 bp are ambiguous within the read.

`-s` `--smallest-sRNA=<arg>`
Minimum length of sRNA to report

`-t` `--largest-sRNA=<arg>`
Maximum length of sRNA to report

`-p` `--permit-missing-adapter`
Permit reads where the adapter cannot be found (not recommended).

4.2.3 shore mapflowcell

This program performs the actual read alignments to a reference genome. `shore mapflowcell` operates on the level of a FlowcellFolder. The basic idea is to specify one FlowcellFolder within a ProjectFolder that should be mapped and all other file handling is performed by SHORE automatically. Sometimes it is desired to re-map only one or a few of the lanes, or read lengths, or maybe just one of the read pairs. The mapping can be restricted to different subsets, with -d -l -t. That way not all reads have to be (re-)mapped if this is needed at some point. Once the mapping is started an empty file will be created, it is either called "mapping.running" or "mapping.finished". No additional mapping can be started if such a file is in the folder. After the mapping of the whole flowcell is finished these files will be removed.

SHORE supports various mapping tools to provide the best option for different applications. The default tool is GenomeMapper and it is extensively tested. Currently the other available options are Vmatch, Novocraft, Bowtie and Eland. If you have a favorite mapping tool that is missing, please contact us and we will check whether it can be included.

Usage:

Mandatory:

`-f` `--files=<arg[,...]>`
Shore folders or files to be mapped. These can be FlowcellFolders, LaneFolders, or single files in `reads_0.fl` format. Folders that already contain a `map.list` file will be skipped.

`-i` `--index-file=<arg[,...]>`
Fasta file(s) in IndexFolder, *.shore file. Multiple files may be specified to do a step-wise alignment using multiple indeces. Note that some downstream analyses like `shore consensus` currently only work with a single index.

Alignment parameters:

`-n` `--edit-distance=<arg>` (Default: '0')
Maximum edit distance. This is the maximum number of mismatches plus gaps allowed in any read alignment.
For a percentage read length based maximum edit distance add a "%".

`-g` `--maxgaps=<arg>` (Default: '0')
Maximum number of gaps (0-n) (only genomemapper, bwa, novoalign, vmatch). Indicates how many of the mismatches are allowed to be gaps rather than plain mismatches. This cannot be higher than number of mismatches specified by `-n`.
For a percentage read length based gap count add a "%".

`-e` `--gapextension=<arg>`
Maximum gap extension (0-n). Only used with bwa:
-g defines max gap openings and -e max extensions per gap opening.

`-q` `--hamming=<arg>`
Quality-weighted hamming distance as defined by MAQ. Only supported for bowtie and novocraft. Overwrites -n and -g. Permitted values: 0 to inf.
`--ed-restriction=<arg>` (Default: 'off')
Automatically restrict the edit distance based on seed size and read length according to the seed-lemma. (off — on — strict)

Parallelization:

`-c` `--cores=<arg>` (Default: '1')
Number of processors/cores. Parallelization of the mapping process.

`-b` `--batch-size=<arg>` (Default: '100000')
Number of reads per thread.

`--no-native-mt`
Always use shore parallelization instead of the mapping tool's own. This will usually speed up the mapping when using bwa or bowtie, while sacrificing a larger amount of RAM.

Mapping strategy:

`-R` `--report=<arg>`
Maximum reported alignments. If reads have more hits to the reference genome than this, then only a random subset of the hits is reported.

GenomeMapper options:

`-a` `--all-hit-strategy`
All hits are reported, even if better matches exist (only within the specified alignment parameters). Slow!

`-2` `--best2-strategy`
Report the best and the second best hit.

`-s` `--seed-threshold=<arg>`
Discard seeds that occur more frequently than this threshold.

`-l` `--seed-length=<arg>`
Discard seed hits smaller than this size.

Mapping tools:

`-v` `--mapper=<arg>` (Default: 'genomemapper')
Specifies the mapping tool. Select from <genomemapper> <vmatch> <novo> <bowtie> <eland> <bwa>.

`-C` `--color`
Indicates that reads and index are in colorspace

Paired end sequencing:

`-p` `--PE`
Paired-end mode. Setting this option prepares the alignments for later use with `shore correct4pe`.

Other:

`-z` `--compress-maplist`
Compress mapping files. Not recommended because some follow up analyses do currently not work with compressed map files.

`-Y` `--nocompress-leftover`
Do not compress leftover files

`--use-seeds=<arg[,...]>`
Use multiple seed lengths if available in index folder.

`--rplot`
Graphical output of statistics using R.

`--stat`
This will cause folders that are skipped because they already contain a `map.list` file to be added to the mapping statistic.

4.2.4 shore correct4pe

`shore correct4pe` finds the most likely mapping of repetitive reads by utilizing paired-end information. While in paired read mapping each read is aligned separately, read pair information can be used to increase the likelihood of an alignment by selecting the paired alignment based on the most likely distance between the pairs.
`shore correct4pe` starts with estimating the insert size distribution. The upper boundary of this distribution is usually very sharp (clones longer than expected seem to be very rare), whereas the lower boundary is not as sharp and very small clones can be observed as well. Therefore only an upper border is required and a lower boundary can be set optionally. This insert size distribution is then translated into a probability distribution for the observation of a given distance of a pairing (where pairing is defined as the combination of one of the mappings of read 1 with one of the mappings of read 2). All mappings of both reads of a pair are pair-wise compared and all pairings with a probability equal zero are dismissed. Mappings which are not in a pairing with a probability above zero are deleted. This removes all repetitive mappings, which resulted from repeats. If there is a mapping of one read pair with two different mappings of the other read the more likely pairing is kept.
If all pairings have zero probability all mappings of both reads are kept. These are the discordant (unhappy) read pairs which typically are used to predict structural variants.

Each lane has to be started separately, (this will be automated in a future release). shore correct4pe will plot the insert size distribution using the R if -p is specified. R has to be installed and included in the $PATH environment variable.

Usage:

Mandatory:

-l INT
Input directory. Has to be a lane folder.

-x INT
The maximal insert size. An estimation distribution of the insert sizes will be calculated. Read pairs which mappings are further distant than x will not be considered for this distribution.

Optional:

-d Delete original map.lists to save memory space, note that the repetitive alignments which are removed by the paired-end correction will be lost. (Most likely there will be no use for them anyway.)

-p
Switch on insert size distribution plotting (default: <not set>)

-n
The lower border of the insert size distribution. Tests have shown that the insert size can be smaller than expected and -n should be handled with care. (default: 0)

4.2.5 shore merge

Every short read sequencing project can be divided into two phases. First comes the read filtering and mapping, and second the analysis of the alignments. In the analysis step, all alignments have to be considered at once. Therefore all map.list files in all FlowcellFolders have to be merged into one file. `shore merge` combines all map.list files into the AlignmentFolder. Based on this file multiple analyses with varying parameters can be run. If the FlowcellFolder includes insert_dist.txt files generated by `shore correct4pe`, these files will also be merged into the AlignmentFolder. Note: After merging, the map.list files are now stored twice on the hard-drives. These files are the largest files in the ProjectFolders, thus it is recommended to delete the map.list in the AlignmentFolder once the analysis is finished.

Usage:

Mandatory:

-p STRING
List of FlowcellFolders to be merged, comma separated.

-d STRING
AlignmentFolder name, will be created

Optional:

-s
Splits merged map.list file into multiple files each representing one chromosome or contig. This can speed up the subsequent analysis steps as each chromosome could be analyzed separately on a different processor or server. Recommended for large genomes (e.g. human).

4.2.6 shore consensus

The common output from whole genome re-sequencing projects are lists of all identified polymorphisms (e.g. SNPs, indels, CNVs) as well as reference-like positions. In addition a consensus sequence or contigs can be generated by combining all high quality predictions. `shore consensus` is providing this functionality by sequentially scanning an alignment to gather all read information available at a specific locus (i.e. called bases, base qualities, coverage, repetitiveness, alignment quality). This information is subsequently used to predict differences to the reference sequence. `shore consensus` can also be used to identify minor alleles (SNPs or short indels) in pooled samples. In addition `shore consensus` estimates several characteristics of a run ahead of the actual consensus calling. This includes min and max read length, min and max mismatches, sequencing depth, observed local repetitiveness and GC content bias. Consensus also provides multiple project statistics regarding sequencing error rate, correlation of quality values to observed errors and coverage biases due to local GC content, which can be used to optimize further analysis (e.g. deletions should not be called in low GC content regions if a strong GC bias is observed). The result file structure of `shore consensus` is explained in more detail in chapter 3.2.

Note: `shore consensus` can be applied to sRNA-seq, mRNA-seq and ChIP-seq data, however SHORE provides more appropriate tools ('coverage' and 'peakup') which will be explained in the next sections.

Usage:

Mandatory:

-n STRING
Arbitrary name (any of species, strain, accession, project or any other ID)

-f STRING
Reference genome sequence from the IndexFolder, *.shore file

-o STRING
AnalysisFolder, will be created.

-i STRING
Merged map.list file created by shore merge.

Quality thresholds:

-q INT
Cutoff for base masking using quality values (default: 5)

-c INT
Cutoff for base masking using chastity values (default: 57)

Base-calling (decision tree approach):

-x INT
Minimum coverage at a position to make a prediction attempt. Regions of lower coverage will not be considered. The value also applies for minimum number of reads supporting a minor allele (SNP or indel) in pooled samples. (default: 3)

-m INT
Maximum observed to expected coverage. (default: 3)

-e DOUBLE
Minimum observed to expected coverage. (default: 0.1)

-y INT
Minimum concordance of base calls at homozygous calls excluding masked bases. (0 to 1) (default: 0.8)

-d DOUBLE
Minimum concordance of homozygous Indels. (default: 0.67)

-t DOUBLE
Minimum allele frequency for heterozygous calls (0 to 1) (default: 0.2)

-u DOUBLE
Minimum allele frequency for minor allele calls in pooled samples. (0 to 1) (default: 0.02)

-z DOUBLE
Maximum base quality supporting the prediction has to be at least z (default: 10)

Basecalling (scoring matrix approach):

-a STRING
Scoring matrix file (activates new basecalling approach)

-b DOUBLE
Minimum allele frequency of alternative base call. (default: 0.2)

Optional:

-s

Consensus analysis using transcriptome (mRNA-seq), ChIP-seq or other non Poisson distributed reads. Uses optimized SNP and indel prediction. Turns off CNV analysis and reduces output size.

-w

Use graph based map.list format (only genomemapper)

-v

Create additional output files also containing all intermediate data. This option provides a large file with more than 70 columns of data for each position of the genome. It can be used to feed custom prediction algorithms. Other predictions only available with this option are: paired end mapping orphans distribution, polymorphic clusters, unsequenced-nonrep and unsequenced-core. All additional data is stored in a subfolder called 'supplementary_data'. This option is mandatory for the add-on SHOREmap!

-r

Graphical output of statistics using R. This is highly recommended if R is installed on the server. The presented plots provide an easy way to analyze the quality of runs and of the analysis. They can also be used to alter quality thresholds (e.g. -q -s -c and -z) to improve analysis results.

-N

Turn off calculation of long deletions, duplications and any other CNVs.

4.2.7 shore structure

`shore structure` enables the detection of diverged regions through clustering of mate pairs alignments with an unexpected distance and/or orientation to each other. Typically the recall is very good for deletions, but insertions longer than the insert size are cannot be revealed. In addition `shore structure` calls inversions. Works for homocygous changes only, currently.

Mandatory:

`-N`

The name of the sequenced sample.

`-i`

A map.list file with all alignments.

`-o`

Output folder, should not exist as it will be created by shore.

`-t`

Translation file mapping the internal chromosome identifier to the one in the original reference file. This file can be found in the IndexFolder, called '*shore.trans'.

`-C`

Annotation of the centromeres. Centromeres attract false mapping leading to huge computation overload and false positive proedictions. This allows to exclude centromeric regions. The file format is straigt-forward. Each line features one centromere by describing three columns, chromosome, begin and end. Each of them tab-delimited.

`-I`

Does not run `shore structure` Instead it outputs mean, stddev, min and max of the given insert distribution file to STDOUT. After wards it stops.

Index and single read mapping:

`-b`
: orphan_distribution.txt file from the ConsensusAnalysis folder created by `shore consensus`

`-c`
: unseq_core.txt file from the ConsensusAnalysis/supplementary_data/ folder created by running `shore consensus` -v.

`-h`
: unseq_core.txt file from the ConsensusAnalysis/supplementary_data/ folder created by running `shore consensus` -v..

Set library parameter:

`-s`
: Expected insert size of sequenced clones.

`-l`
: Library identifier as defined with `shore correct4pe`

`-T`
: Library type, paired-end or mate pair, set to PE or MP.

Set library parameter:

`-n` default is 2
: Minimum number of pairings per cluster.

`-u`
: Consider unhappy mates only (for small SVs set to 0). Setting to 0 will increase the recall rate, as it calls small SVs aswell but will increase run-time drastically.

Seed clustering:

`-n` default is 2
: Minimum number of pairings per cluster.

`-u`
: Consider unhappy mates only (for small SVs set to 0). Setting to 0 will increase the recall rate, as it calls small SVs aswell but will increase run-time drastically.

Cluster affinity calculation:

`-a` default is 0
: Cluster affinity threshold. Changing this will cluster less read pairs, this will break up SVs. Just for testing.

`-m` default is 3
: Restrict the insert size distribution and ignore outliers.

`-q` default is 500000.
: Maximal size for deletions.

`-g` default is 3 for PE libs and 50 for MP lids.
: Granularity for insert size calculations. Setting to 1 will increase runtime but estimations of indel length will be more accurate.

Clonal artifacts:

`-r`
: Set to remove clonal duplicates within the sequencing lib.

Clonal artifacts:

`-e`
: Exclusion of chromosomes.

`-E`

Exclusion of read ids. Simple line separated files with read ids listed.

`-O`
Exclude particular regions. We use oversampled.txt from the ConsensusAnalysis/supplementary_data/ created by `shore consensus` -v.

`-k` default is 0
Extension of the regions given with -O.

Coverage in deleted regions:

`-y` default is 1.0
Maximal fraction of core alignments overlapping deletion. As reptitive alignments are allowed to reside in deletions even in homocygous samples this value is set to 1.0.

`-A` default is 0.9
Maximal fraction of non-repetitive core alignments overlapping deletion. That is 10 percent of the deletion need to be gone completely.

Coverage in deleted regions:

`-B` default is 1000
Maximal number of pairs supporting an indel.

`-f` default is 3
Minimal number of pairs supporting an indel.

`-D` default is 1000
Maximal number of pairs supporting an inversion.

`-j` default is 3
Minimal number of pairs supporting an inversion.

4.2.8 shore coverage

For analysis of expression levels of mRNAs and small RNAs or for detection of unknown transcripts it is typically required to generate a coverage graph and to define expressed segments based on consecutive coverage. In case of mRNA-Seq, this coverage graph can be further enhanced by accurate splice-site prediction using reads overlapping with splice junctions. `shore coverage` generates a coverage graph by sequentially scanning the alignment and basically counting reads.

Usage:

Input:

`-m` `--mapfiles=<arg[,...]>`
A comma separated list of map.list files or flowcell folders. Separate coverage and segmentation files will be generated for each input file. These files may also be given without an option switch at the end of the command line. Files or folders separated by a ':' will be treated like a single file.

`-n` `--merge-input`
Merge all input files. This is a shortcut option that will behave like all files given in the `-m` option were separated by ':'.

Output:

`-o` `--output-directory=<arg>` (Default: 'CoverageAnalysis')
Output directory (will be created)

`-s` `--segmentation`

Write segmentation files

`-t` `--merge-segments=<arg>`
Overlap in base pairs for merging segment files (may be negative to allow gaps); if unspecified, segments will not be merged

`-q` `--no-coverage`
Do not write coverage files. This may be used if one is only interested in the segmentation results.

`-z` `--compress`
Compress output files

`--rplot`
Only works together with the `R` option. This will plot the coverage graph for the range specified in `R` using R. A pdf file will be generated in the output directory.

`--ylim=<arg>`
Set y axis limit for plots (default: auto)

`--phasing=<arg>`
Visualize <arg>-mer phasing. This will include a 'phasing clock' graph in the pdf file.

Flowcell folders:

`-K` `--lanes=<arg[,...]>`
Specifies which lane folders to use (default: all)

`-L` `--lengths=<arg[,...]>`
Specifies which read length folders to use (default: all)

Read filtering:

`-H` `--hits-range=<arg>`
Set the allowed range of repetitiveness ('1,1' = nonrep reads)

`-M` `--mm-range=<arg>`
Set the allowed range of mismatches

`-R` `--pos-range=<arg>`
Only use reads that overlap with the range [chr1:pos1,chr2:pos2)

`--assume-length=<arg>` (Default: '400')
Assume maximal read length <arg>, enables fast range query in uncompressed files

`-X` `--p3fix=<arg>`
Set the 3' end to a fixed distance from the 5' end

`-N` `--read-lengths=<arg[,...]>`
Use only reads of the given length(s)

`-T` `--strand=<arg>`
Use only reads from the given strand

`-B` `--doublets=<arg>`
Report at maximum <arg> reads with the 5' end at the same position on the same strand

Coverage:

`-W` `--weight-repetitive=<arg>` (Default: 'divide')
How to weight repetitive hits (divide — multiply — const). By default, a read mapping equally well to e.g. four locations is given a score of 0.25.

Segmentation:

`-C` `--static-threshold=<arg>` (Default: '10')

Coverage threshold [>] for static segmentation. Every region where the coverage is above this threshold will be regarded a 'segment', everything else a gap.

`-J` `--minsize=<arg>` (Default: '20')
Segment size threshold [>=] for static or dynamic segmentation. Any segment smaller than this size will not be reported.

`-V` `--probation=<arg>` (Default: '0')
Allow a mitigated coverage threshold for at most <arg> base pairs inside a segment. This mitigated threshold will be calculated using the value specified in `--mitigator` together with either `-C` or `-D`.

`-Q` `--mitigator=<arg>` (Default: '1')
Modifier for calculation of the mitigated threshold. The coverage threshold will be *multiplied* for static segmentation, whereas for the dynamic segmentation algorithm, the poisson threshold will be *divided* by this value to calculate the weaker threshold. for the `--probation` option.

`-D` `--dynamic`
Switches to the dynamic segmentation algorithm. This will dynamically adjust the coverage threshold the local background 'noise'.

`-S` `--window-size=<arg>` (Default: '2000')
Sliding window size for dynamic segmentation

`-P` `--poisson-threshold=<arg>` (Default: '0.05')
Poisson probability threshold [<=] for dynamic segmentation

4.2.9 shore peak

`shore peak` provides enriched region prediction for ChIP-Seq experiments. Significance of the predicted regions is assessed by comparison to the specified control samples.

Replicate experiments may be processed simultaneously by specifying multiple experiment and control paths. While the significance of each peak region is then tested for independently for each replicate, the region prediction itself is performed jointly for all experiments to obtain results that are immediatly comparable.

Usage:

Mandatory:

`-o` `--outfolder=<arg>` (Default: 'PeakAnalysis')
Output folder (will be created)

`-i` `--chip-paths=<arg[,...]>`
ChIP experiment map files / flowcell folders. Replicate experiments must be comma-separated. Multiple paths may be specified to belong to the same experiment by separating them by a colon.

`-c` `--ctrl-paths=<arg[,...]>`
Control experiment map files / flowcell folders. As with `--chip-paths`, multiple paths may be specified.

Segmentation:

`-S` `--window-size=<arg>` (Default: '2000')
Sliding window size for dynamic segmentation

`-P` `--poisson-threshold=<arg>` (Default: '0.05')

Poisson probability threshold [<=] for dynamic segmentation

`-V` `--probation=<arg>` (Default: '0')
Allow a mitigated threshold for at most <arg> base pairs inside a segment

`-Q` `--mitigator=<arg>` (Default: '1')
Modifier for calculation of the mitigated threshold

`-J` `--minsize=<arg>` (Default: '131')
Segment size threshold [>=]

Normalization:

`-b` `--binsize=<arg>` (Default: '400')
Size of read bins for normalization. This size should be increased for very low-coverage samples to still allow a valid estimation of the background coverage.

`-q` `--rankmaxquant-ubound=<arg>` (Default: '1')
Quantile upper bound for the rank maxima of the bins used. Setting this to a value < 1 will force a less conservative control sample normalization.

Flowcell folders:

`-K` `--lanes=<arg[,...]>`
Specifies which lane folders to use (default: all)

`-L` `--lengths=<arg[,...]>`
Specifies which read length folders to use (default: all)

Read filter:

`-H` `--hits-range=<arg>`
Set the allowed range of repetitiveness. Set to '1,1' to only use uniquely mapping reads, which is currently the most common choice for ChIP-Seq analyses.

`-M` `--mm-range=<arg>`
Set the allowed range of mismatches

`-R` `--pos-range=<arg>`
Only use reads that overlap with the range [chr1:pos1,chr2:pos2)
`--assume-length=<arg>` (Default: '400')
Assume maximal read length <arg>, enables fast range query in uncompressed files

`-X` `--p3fix=<arg>` (Default: '130')
Set the 3' end to a fixed distance from the 5' end. This value should roughly match the DNA fragment size that is estimated for the experiment.

`-N` `--read-lengths=<arg[,...]>`
Use only reads of the given length(s)

`-B` `--duplicates=<arg>`
Report at maximum <arg> reads with the 5' end at the same position on the same strand. Usually, setting `-F` should be the better choice (this is the default).

`-F` `--poissonifier-width=<arg>` (Default: '13')
Set the window size for the adaptive duplicate read filter (set to zero to disable)

Peak filtering:

`-n` `--nsigma=<arg>` (Default: '6')
Allow the mean segment coverage any control sample to be at most <arg> std. deviations higher than the median before discarding the segment
`--min-xshift=<arg>` (Default: '10')
Require a certain shift for the reverse strand peak in at least one experiment
`--min-foldchange=<arg>` (Default: '2')

Require a minimum normalized fold change of <arg> for experiment vs. control for at least one experiment

Other:

`--paranoid`
Calculate a more conservative FDR. Should give a more realistic estimation for samples that are heavily amplified by PCR.

`-d` `--rankproduct=<arg>` (Default: '10000')
Number of simulations for rankproduct PFP estimation (set to zero to disable PFP estimation). The rankproduct PFP is extremely conservative if the number of replicates is low and can only be used to identify a few very high confidence peaks. This is only used if replicate experiments are specified.

`--rplot=<arg>` (Default: '100')
Plot the first <arg> peaks using R

`-r` `--index-file=<arg>`
Extract sequence information for each segment from *.shore index file

`-a` `--annotation-file=<arg>`
Annotation file. This is a file in standard GFF format. Overlapping, upstream and downstream genes will be reported for each enriched region.

`-O` `--chr-ordering=<arg[,...]>`
Allows to specify the order of chromosome entries in the annotation file

`--so-filter=<arg[,...]>` (Default: 'gene,transposable_ element_gene')
Only parse toplevel annotation features of the given SO types

4.2.10 shore srna

The purpose of `shore srna` is to facilitate the analysis of small RNA sequencing data. The genome is scanned for regions where significant amounts of small RNAs are expressed and annotates these loci by the sRNA size that predominates.

Usage:

Mandatory:

`-s` `--samples=<arg[,...]>`
Shore folders (comma-separated; colon-separated items will be treated as a single assay)

`-o` `--outfolder=<arg>` (Default: 'SrnaAnalysis')
Output folder

Segmentation:

`-C` `--static-threshold=<arg>` (Default: '10')
Coverage threshold [>]

`-J` `--minsize=<arg>` (Default: '15')
Segment size threshold [>=]

`-V` `--probation=<arg>` (Default: '0')
Allow a mitigated threshold for at most <arg> base pairs inside a segment

`-Q` `--mitigator=<arg>` (Default: '1')
Modifier for calculation of the mitigated threshold

`-v` `--overlap=<arg>` (Default: '1')
Required overlap for merging segments (may be negative to allow gaps)

Flowcell folders:

`-L  --lengths=<arg[,...]>`
Specifies which read length folders to use (default: all)

Read filter:

`-H  --hits-range=<arg>`
Set the allowed range of repetitiveness ('1,1' = nonrep reads)

`-M  --mm-range=<arg>`
Set the allowed range of mismatches

`-R  --pos-range=<arg>`
Only use reads that overlap with the range [chr1:pos1,chr2:pos2)

`--assume-length=<arg>` (Default: '400')
Assume maximal read length <arg>, enables fast range query in uncompressed files

4.2.11 shore count

`shore count` calculates the read count as well as other properties for regions in the genome that have already been defined by some other means. It may be used to analyse either fixed-size 'jumping windows' over the genome or regions defined in an input file, e.g. to analyse annotated coding regions or to re-analyse regions defined by the segmentation algorithms of `shore coverage`, `shore peak` or `shore srna`.

Accepted input file formats are either tab-delimited plain text files with a header specifying the columns 'chr', 'pos' and 'size', or a standard GFF file.

Usage:

Mandatory:

`-m  --mapfiles=<arg[,...]>`
Map files or flowcell folders (comma-separated; colon-separated items will be treated as a single assay)

`-o  --output-folder=<arg>` (Default: 'SegmentAnalysis')
Output folder, will be created

Variable size:

`-f  --segment-file=<arg>`
Set file with segment information (expects a sorted file with columns chr, pos, size, strand)

Fixed size:

`-s  --segment-size=<arg>`
Use segments of fixed size <arg> instead of a file

`-j  --segment-distance=<arg>`
Distance of fixed size segments (defaults to segment size)

`-t  --strand-specific`
Count both strands separately

Output:

`-a  --fasta-file=<arg>`
If a fasta file is provided, the sequence will be reported for each segment

Counting:

`-O` `--overlap=<arg>` (Default: '50%')
Required amount of overlap between read and feature (percentage or absolute)

`-W` `--weight-repetitive=<arg>` (Default: 'divide')
How to weight repetitive hits (divide — multiply — const)

Flowcell folders:

`-K` `--lanes=<arg[,...]>`
Specifies which lane folders to use (default: all)

`-L` `--lengths=<arg[,...]>`
Specifies which read length folders to use (default: all)

Read filter:

`-H` `--hits-range=<arg>`
Set the allowed range of repetitiveness ('1,1' = nonrep reads)

`-M` `--mm-range=<arg>`
Set the allowed range of mismatches

`-R` `--pos-range=<arg>`
Only use reads that overlap with the range [chr1:pos1,chr2:pos2)
`--assume-length=<arg>` (Default: '400')
Assume maximal read length <arg>, enables fast range query in uncompressed files

`-X` `--p3fix=<arg>`
Set the 3' end to a fixed distance from the 5' end

`-N` `--read-lengths=<arg[,...]>`
Use only reads of the given length(s)

`-T` `--strand=<arg>`
Use only reads from the given strand

`-B` `--doublets=<arg>`
Report at maximum <arg> reads with the 5' end at the same position on the same strand

4.2.12 shore binom_test

`shore binom_test` can be used to evaluate two sets of count data agaist each other using a binomial test.

Usage:

Allowed options:

`-i` `--input-file=<arg>` (Default: 'stdin')
Read count input file

`-o` `--output-file=<arg>` (Default: 'stdout')
Output file

`-p` `--distribution-p=<arg>` (Default: '0.5')
Parameter p of the binomial distribution

`-n` `--normalization-file=<arg>`
File with scaling factors for each column (overrides '-p')

`-a` `--alternative=<arg>` (Default: 'less')
Specifies the alternative hypothesis for the test (less — greater — twosided)

`-1` `--first-column=<arg>`
Name of the first read count column
`-2` `--2nd-column=<arg>`
Name of the second read count column (tested vs. column 1)
`-j` `--input-header=<arg[,...]>`
Specify header for input file, if not available
`-f` `--fold-change`
Report fold enrichment values
`--fdr-bh`
Calculate Benjamini-Hochberg FDR
`--sort`
Sort output by p-value/FDR

4.2.13 shore mtc

The subprogram `shore mtc` implements various different multiple testing correction methods. The expected input is a tab-delimited text file with a header, and the column containing the p-values to be adjusted must be named `raw_p`.

Implemented methods include Benjamini-Hochberg false discovery rate control (fdr_bh), Bonferroni familywise error rate control (fwer_bonferroni), Holm familywise error rate control (fwer_holm), Hochberg familywise error rate control (fwer_hochberg), Sidak singlestep familywise error rate control (fwer_sidak_ss), Sidak stepdown familywise error rate control (fwer_sidak_sd) and Benjamini-Yekutieli false discovery rate control (fdr_by).

Usage:

Mandatory:
`-m` `--method=<arg[,...]>`
Select correction method(s), out of:
fdr_bh, fwer_bonferroni, fwer_holm, fwer_hochberg, fwer_sidak_ss, fwer_sidak_sd, fdr_by
`-i` `--input-file=<arg>` (Default: 'stdin')
The file the raw p-values are read from (expects a column 'raw_p')
`-o` `--output-file=<arg>` (Default: 'stdout')
Output file

Output:
`-u` `--fdr-max=<arg>` (Default: '1')
Maximum q-value to report
`-e` `--echo-comments`
Echo all comments read from input files to stdout
`-q` `--quiet`
Do not print input, only report the q-values

Other:
`-j` `--input-header=<arg[,...]>`
Use arg as input file header

4.2.14 shore ranksim

`shore ranksim` ranks data in two or more tab-delimited text files using the specified comparators and then calculates the rankproduct percentage false positives for each item.

Usage:

Allowed options:

`-o` `--outfile=<arg>` (Default: 'stdout')
Output file

`-k` `--key-cmp=<arg>`
Comparator string for the unique row identifiers (Concatenation of key types and column ids (counted from 1). Valid key types: t (text), i (integer) and f (float); capital letters reverse the sort order - e.g. '-k i1t5i3I7')

`-d` `--data-cmp=<arg[,...]>`
Comparator strings for the data that will be ranked

`-n` `--nsim=<arg>` (Default: '1000')
Number of simulations

`-s` `--sort`
Sort output

`--no-pfp`
Do not estimate PFP

`--ranksum`
Use rank sums instead of rank products

4.2.15 shore annotate

`shore annotate` can be used to annotate previously defined genomic regions with the overlapping or nearest genes present in an annotation file. Only the central base of each region will be annotated. The annotation file must be in standard GFF format.

Usage:

Allowed options:

`-a` `--annotation-file=<arg>`
Annotation file

`-O` `--chr-ordering=<arg[,...]>`

`-f` `--feature-file=<arg>`
File with the features to be annotated

`-o` `--outfile=<arg>` (Default: 'stdout')
Output file

`--gff`
Write output in GFF format

`--so-filter=<arg[,...]>`
Only parse toplevel features of the given SO types A useful setting with typical GFF annotations is e.g. `gene,transposable\_element\_gene`

`--print`
Just print the annotation tree

`--query-pos=<arg>`

Query annotation for the given position

4.2.16 shore convert

`shore convert` provides several format converters to allow the integration of other mapping and analysis tools, as well as widely used genome browsers (GBrowse, UCSC). The input format of SHORE (*.fl) can be converted in both directions to fasta, fastq, qual. The alignment format of SHORE (map.list) can be converted in both directions to GFF, BED, the vmatch format and the Novocraft native format. Result files from `shore consensus` and `shore coverage` can be converted to GFF and BED. In addition convert allows the creation of mysql and postgres database tables to load the result files into. Please note that not all of the described converters are activated in the beta-version due to insufficient testing. For a detailed description of all supported converters please see the help page of 'shore convert'

Usage:

Mandatory:

`-c` `--converter=<arg>`
One of the following converters:
- Fastq2Fasta
- Fastq2Qual
- Flat2Qual
- Flat2Fasta
- Flat2Fastq
- Solid2Fastq
- Solid2Flat
- ColorFlat2Fastq
- Maplist2Aln
- Maplist2BED
- Maplist2GFF
- Maplist2Eland
- Maplist2Flat
- Maplist2Sam
- NovoNative2Maplist
- Bowtie2Maplist
- Eland2Maplist
- Sam2Maplist
- Contig2AFG
- HomozygousSNP2GFF
- HomozygousSI2GFF
- ExpandTabs
- Pair2Fastq

`-i` `--input-file=<arg>` (Default: 'stdin')
Source for the converter. Note that some converter require additional input files specified with e.g. `-2`

`-o` `--output-file=<arg>` (Default: 'stdout')
Destination for the converter. Note that 'Flat2Qual' does not require an output file. Instead 2 files are created called <Input-file>.fa and <Input-file>.qual.

Eland2Maplist converter:

`-r  --reads=<arg>`
Reads in SHORE flat file format

Eland2Maplist & Sam2Maplist converter:

Contig2AFG converter:
`-a  --assembler=<arg>` (Default: 'all')
Select contigs only from this assembler
`-s  --cut-size=<arg>` (Default: '1000')
Cut size for contigs
`-m  --min-size=<arg>` (Default: '100')
Minimum contig length
`-x  --min-overlap=<arg>` (Default: '0')
Minimum kmer overlap length
`-y  --max-overlap=<arg>` (Default: '999')
Maximum kmer overlap length

Pair2Fastq converter:
`-2  --reads2=<arg>`
Second reads file

4.2.17 shore sort

`shore sort` allows to access SHORE's internal sort program from the command line.

Usage:

Allowed options:
`-i  --infiles=<arg[,...]>`
A comma-separated list of input text files
`-o  --outfile=<arg>` (Default: 'stdout')
Output file
`-k  --keystring=<arg>`
Concatenation of key types and column ids (counted from 1). Valid key types: t (text), i (integer) and f (float); capital letters reverse the sort order - e.g. '-k i1t5i3I7'.
`-t  --tmpdir=<arg>`
Temporary file directory (defaults to $TMPDIR or /tmp)
`-b  --blocksize=<arg>` (Default: '2048')
Block size in megabytes
`-m  --nur-merge`
Merge already sorted files
`-u  --upper-bound=<arg>`
Returns byte offset (counted from 0) and text of the first line in a sorted file that compares greater than the text given in ¡arg¿
`-C  --no-comments`
Do not treat line comments and empty lines specially

5 SHORE Roadmap

SHORE is still subject to change, but we try to keep the folder structure and the file formats as stable as possible to ensure backward compatibility. Current developments include:

SHORErna:

- Spliced alignments using GenomeMapper/QPalma
- Quantitative analysis of mRNA-Seq and sRNA-Seq
- Detection of differentially expressed loci in multiple samples

SHOREpeak:

- Motif search in peak segments
- Improved peak ranking

SHOREassembly:

- Whole genome homology-guided assemlby (WGHA)
- Targeted homology-guided assemlby (THA)
- Low coverage assembly
- Scaffolding

SHOREvariant:

- Improving structural variant and CNV detection using mate pairs
- Improving SNP and indel detection for SOLiD

Misc:

- New machine learning based trimming algorithm RTrim
- Inclusion of additional alignment tools
- Additional and improved converters
- Optimized data exchange with Polymorph viewer

We appreciate any kind of feedback. Please do not hesitate to contact us in any cases of problems or questions.

Stephan Ossowski <stephan.ossowski@tuebingen.mpg.de>
Korbinian Schneeberger <korbinian.schneeberger@tuebingen.mpg.de>
Felix Ott <felix.ott@tuebingen.mpg.de>
Jörg Hagmann <joerg.hagmann@tuebingen.mpg.de>
Sebastian Bender and Alf Scottland