

# SSRG — A SIMPLE PIPELINE TO ASSESS GENETIC DIVERSITY BETWEEN BACTERIAL GENOMES

SSRG MANUAL – VERSION 0.7

DECEMBER 12, 2019

Pombert lab, Illinois Tech

## TABLE OF CONTENTS

<b>Overview .....</b>	<b>2</b>
<b>Workflows .....</b>	<b>2</b>
<b>Dependencies .....</b>	<b>4</b>
<b>Installation .....</b>	<b>4</b>
<b>Command line usage .....</b>	<b>5</b>
Quick reference – Read mapping.....	5
Quick reference – Genetic distance estimation.....	5
Detailed options.....	5
<i>Read mapping</i>	
queryNCBI.pl                      Downloads data automatically from GenBank.....	5
SSRG.pl                              Synthetic Short Read Generator .....	6
get_SNPs.pl                        Read mapping/variant calling .....	6
bam2fastq.pl                      Extract reads in FASTQ format from BAM files.....	8
sort_stats.pl                      Table generator .....	8
synonymy.pl                        Sorts synonymous/non-synonymous SNPs .....	8
count_SNPs.pl                     Quick metrics .....	9
<i>Genetic distances</i>	
run_Mash.pl                        Perl wrapper for MASH.....	9
MashToDistanceMatrix.pl        Generates distance matrices from MASH outputs.....	9
MashR_plotter.pl                 Plots distance matrices .....	10
<b>REFERENCES .....</b>	<b>12</b>

## OVERVIEW

The SSRG pipeline was created as a simple, focused tool to investigate SNPs between prokaryote genomes. The pipeline uses the common SNP-calling approach of read-mapping against references, standardizes experimental conditions for more accurate SNP comparisons, and integrates ubiquitous methodologies for both analysis and visualization. The unique feature of the SSRG pipeline resides in the creation of synthetic short reads from complete or draft genomes, which can then be fed to the read mapping/variant calling tools. Note that this approach works only for haploid genomes. Alternatively, users can also select any compatible FASTQ datasets to use with the pipeline.

Assessing the genetic diversity between genomes often involves the calculation of single nucleotide polymorphisms (SNPs) and insertions/deletions (indels). This is usually done by mapping short accurate sequencing reads from one or more species against a reference genome, from which variants are called. This approach works well when short read data from published genomes are available in public repositories, which is not always the case, especially now that bacterial genome sequencing is shifting towards the use of long read technologies. While genomes and/or long reads can be aligned against each other, the results are often suboptimal when the investigated chromosomes are highly reorganized, which can cause the mapping to fail. A simple solution to this problem is to deconstruct the genomes or long reads into shorter fragments, a shotgun approach, and to use these smaller synthetic reads as input for mapping.

Deconstructing genomes into synthetic reads has the following advantages:

- This approach allows the comparisons of genomes for which sequencing read datasets are not available in public repositories.
- This approach helps standardize datasets by providing reads with the exact same parameters. For example, genomes generated from illumina, PacBio and/or Nanopore data can now be compared without fuss.
- Because bases from complete or draft genomes have been queried multiple times by the sequencing depth, the underlying confidence in the base being called is thus higher than from a single sequencing read. This in turn leads to fewer false positives caused by sequencing errors.

The SSRG pipeline currently can:

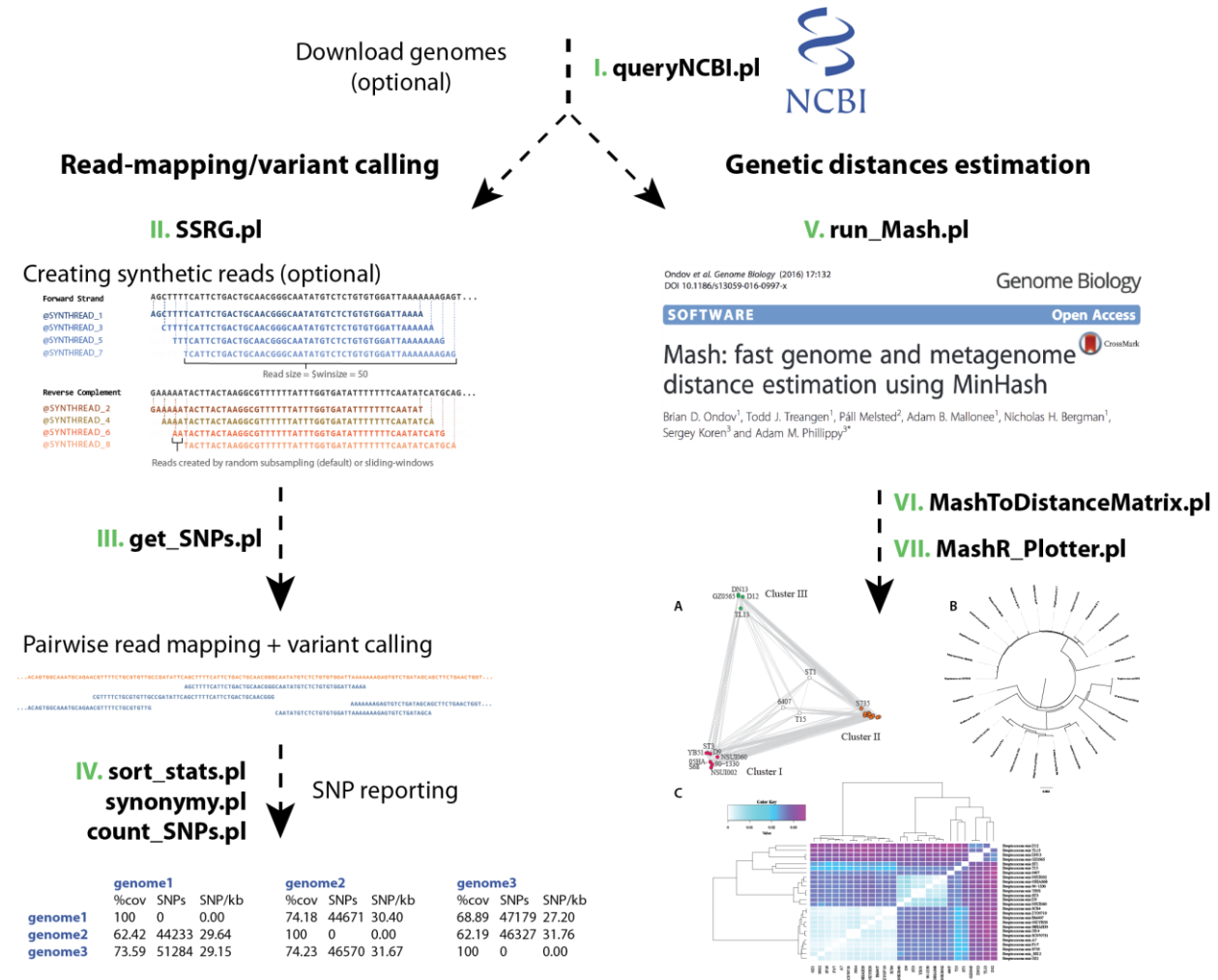
- 1) Download genomes automatically from NCBI using a CSV/Tab-delimited list of desired operational taxonomic units (OTU)
- 2) Calculate pairwise SNPs between FASTQ sequences and reference genomes using standard read mapping approaches.
- 3) Run Mash [1] (<https://github.com/marbl/Mash>; Ondov et al. 2016. DOI: 10.1186/s13059-016-0997-x) and plot the estimated genetic distances as heatmaps, neighbor-joining trees, or clusters (using dimensionality reduction techniques).

## WORKFLOWS

The SSRG pipeline features two independent workflows:

- I. Read-mapping/variant calling
- II. Genetic distances estimation

Users interested in point mutations should use the read-mapping/variant calling workflow. Users only interested in genetic distances should use the genetic distances estimation workflow. This workflow is based on MASH, an excellent and fast tool developed by Ondov *et al.* [1] [Ondov BD, Treangen TJ, Mallonee AB, Bergman NH, Koren S, Phillippy AM (2016) Fast genome and metagenome distance estimation using MinHash. *Genome Biol* 17:132. DOI: 10.1186/s13059-016-0997-x]. The Mash workflow does not identify point mutations.



## DEPENDENCIES

### ## General

- Perl 5 <https://www.perl.org/>

### ## Read mapping (RM)/variant calling (VC)

## At least one RM and VC tools are required for SNP calling, others RM and VC are optional

- Samtools version 1.3.1+ <http://www.htslib.org/>
- BWA (RM) <http://bio-bwa.sourceforge.net/>
- Bowtie2 (RM) <http://bowtie-bio.sourceforge.net/bowtie2/index.shtml>
- HISAT2 (RM) <https://ccb.jhu.edu/software/hisat2/index.shtml>
- Minimap2 (RM) <https://github.com/lh3/minimap2>
- NGMLR (RM) <https://github.com/philres/ngmlr>
- Bcftools 1.3.1+ (VC) <http://www.htslib.org/>
- FreeBayes (VC) <https://github.com/ekg/freebayes>
- VarScan2 (VC) <https://github.com/dkoboldt/varscan>
- Java (for VarScan) <https://www.java.com/>

### ## Genetic distance estimations

- R <https://www.r-project.org/>
- Mash <https://github.com/marbl/Mash>

## INSTALLATION

### *On Fedora/Red Hat*

```
sudo dnf install perl R bwa boost boost-devel zlib zlib-devel gsl gsl-devel autoconf automake \
java-1.?.?-openjdk java-1.?.?-openjdk-devel
```

### *Downloading/installing from GitHub*

1. git clone --recursive <https://github.com/PombertLab/SNPs.git>
2. chmod a+x SNPs/\*.pl; chmod a+x SNPs/\*/\*.pl
3. Install the scripts in your \$PATH (e.g. by adding to your ~/.bash\_profile). To install for all users, you can create a shell script in /etc/profile.d/ on most Linux systems: e.g.  
sudo export PATH=\$PATH:/path/to/SNPs" >> /etc/profile.d/SSRG.sh;\n  
sudo export PATH=\$PATH:/path/to/SNPs/SSRG/" >>/etc/profile.d/SSRG.sh;\n  
sudo export PATH=\$PATH:/path/to/SNPs/MASH/" >> /etc/profile.d/SSRG.sh;\n  
sudo export PATH=\$PATH:/path/to/SNPs/Tools/NCBI/" >> /etc/profile.d/SSRG.sh;\n  
\* Replace /path/to/SNPs by your installation directory

### OPTIONAL

1. If desired, update the VarScan default location in the corresponding line in get\_SNPs.pl,  
i.e. **my \$varjar = '/opt/varscan/VarScan.v2.4.3.jar';**  
## This can also be changed from the command line with the -var option

2. If dependencies are not installed in your \$PATH, you can alternatively insert the installation directories (e.g. \$samtools = '/opt/samtools-1.3.1/bin/') at the top of get\_SNPs.pl to reflect your settings.

### **Installing R packages dependencies**

To install R packages for all users, type the following:

- 1) sudo R
- 2) install.packages c("Rcpp", "gplots", "ggplot2", "ggfortify", "RColorBrewer", "plotly", "ape", "Rtsne")
- 3) quit()

## COMMAND LINE USAGE

### **QUICK REFERENCE – READ MAPPING**

- I. get\_SNPs.pl -fa \*.fasta -fq \*.fastq ## Performs read mapping/variant calling

#### **## Optional**

- I. queryNCBI.pl -fa -l genome\_list.csv ## Downloads genomes from NCBI
- II. SSRG.pl -f \*.fasta ## Creates FASTQ reads for each genome
- III. sort\_stats.pl -s \*.stats -o table\_name ## Creates a tab-delimited table
- IV. synonymy.pl -fa reference.fasta -gff reference.gff -vcf \*.vcf  
## sorts syn/non-synonymous SNPs

### **QUICK REFERENCE – GENETIC DISTANCE ESTIMATION**

- I. run\_Mash.pl -f \*.fasta -o Mash.txt ## Runs MASH
- II. MashToDistanceMatrix.pl Mash.txt ## Converts the output of MASH to  
## a distance matrix
- III. MashR\_plotter.pl -i Mash.csv -o image\_01 ## Plots the distance matrix using R

### **DETAILED OPTIONS**

#### **queryNCBI.pl Downloads data automatically from GenBank**

This script downloads data from the GenBank database using as input a TAB/CSV-delimited list generated from NCBI's Genome Assembly and Annotation reports. For example;

- 1) go to <http://www.ncbi.nlm.nih.gov/genome/genomes/159?>
- 2) click on the Download Table link in the upper right corner of the webpage

### **USAGE**

queryNCBI.pl -l genome\_list.csv [OPTIONS]

### **OPTIONS**

- l (--list) TAB/CSV-delimited list
- fa (--fasta) Retrieve fasta files
- gb (--genbank) Retrieve GenBank annotation files

-p (--protein) Retrieve protein sequences  
-cds Retrieve protein coding sequences

### **SSRG.pl** Synthetic Short Read Generator

This script deconstructs draft or complete genomes into separate sets of FASTQ reads of desired length (default 100). This is particularly useful to perform standardized read mapping analyses and to palliate for missing short read data.

#### **USAGE**

SSRG.pl [options] -f \*.fasta

#### **EXAMPLE (simple)**

SSRG.pl -f \*.fasta

#### **EXAMPLE (advanced)**

SSRG.pl -f \*.fasta -r 250 -i 350 -s 20 -m rand -c 25

#### **OPTIONS**

-v (--version)	Display SSRG.pl version number
-f (--fasta)	Fasta/multifasta input file
-r (--readsize)	Synthetic reads size [default: 100]
-m (--mode)	Sliding windows or random selection (slide or rand) [default: rand]
-t (--type)	Single or paired ends (se or pe) [default: pe]
-i (--insert)	Insert size for pe [default: 250]
-s (--sdev)	Insert size standard deviation percentage [default: 10]
-c (--coverage)	Set sequencing depth [default: 50]
-qs (--qscore)	Quality score associated with each base [default: 30]
-q64	Used the old Illumina Q64 FastQ format instead of the default Q33 Sanger/Illumina 1.8+ encoding

### **get\_SNPs.pl** Read mapping/variant calling

This script performs read mapping between FASTQ datasets and reference genomes in FASTA format using user-selectable read mappers and variant callers. This script also generates read mapping and coverage statistics.

#### **USAGE**

perl get\_SNPs.pl [options]

#### **EXAMPLE (simple)**

get\_SNPs.pl -fa \*.fasta -fq \*.fastq

#### **EXAMPLE (advanced)**

get\_SNPs.pl --fasta \*.fasta --fastq \*.fastq --mapper bowtie2 --caller varscan2 --var ./VarScan.v2.4.3.jar \  
--threads 16

#### **EXAMPLE (paired ends)**

get\_SNPs.pl --fasta \*.fasta --pe1 \*R1.fastq --pe2 \*R2.fastq --X 1000 --mapper bowtie2 --caller freebayes \  
--threads 16

### EXAMPLE (read-mapping only)

```
get_SNPs.pl --fasta *.fasta --pe1 *R1.fastq --pe2 *R2.fastq --X 1000 --mapper bowtie2 --rmo \
--bam --threads 16
```

### OPTIONS

-h (--help)      Display the list of options  
-v (--version)    Display the script version

#### ## Mapping options

-fa (--fasta)                      Reference genome(s) in fasta file  
-fq (--fastq)                      Fastq reads (single ends) to be mapped against reference(s)  
-pe1                                Fastq reads #1 (paired ends) to be mapped against reference(s)  
-pe2                                Fastq reads #2 (paired ends) to be mapped against reference(s)  
-mapper                            Read mapping tool: bwa, bowtie2, minimap2, ngmlr or hisat2  
                                    [default: bowtie2]  
-threads                            Number of processing threads [default: 16]  
-bam                                Keeps BAM files generated  
-sam                                Keeps SAM files generated; SAM files can be quite large  
-rmo (--read\_mapping\_only)        Do not perform variant calling; useful when only interested in bam/sam  
                                    files and/or mapping stats  
-ns (--no\_stats)                    Do not calculate stats; stats can take a while to compute for large  
                                    eukaryote genomes

#### ## Mapper-specific options

-X                                  BOWTIE2 - Maximum paired ends insert size [default: 750]  
-preset                            MINIMAP2 - Preset: sr, map-ont, map-pb or asm20 [default: sr]  
-algo                              BWA - Mapping algorithm: bwasw, mem, samse [default: mem]

#### ## Variant calling options

-caller                            [default: varscan2]      ## Variant caller: varscan2, bcftools or freebayes  
-type                              [default: snp]            ## snp, indel, or both  
-ploidy                            [default: 1]              ## FreeBayes/BCFtools option; set ploidy (if needed)

#### ## VarScan2 parameters (see <http://dkoboldt.github.io/varscan/using-varscan.html>)

-var                                [default: /opt/varscan/VarScan.v2.4.3.jar]      ## Which varscan jar file to use  
-mc (--min-coverage)              [default: 15]            ## Min. read depth at a position to make a call  
-mr (--min-reads2)                [default: 5]             ## Min. supporting reads to call variants  
-maq (--min-avg-qual)              [default: 28]            ## Minimum base quality to count a read  
-mvf (--min-var-freq)              [default: 0.7]           ## Minimum variant allele frequency threshold  
-mhom (--min-freq-for-hom)        [default: 0.75]          ## Minimum frequency to call homozygote  
-pv (--p-value)                    [default: 1e-02]        ## P-value threshold for calling variants  
-sf (--strand-filter)               [default: 0]             ## 0 or 1; 1 ignores variants with >90% support  
                                    ## from a single strand

## **bam2fastq.pl**

This script extracts sequencing reads in FASTQ format from BAM alignment files.

### **USAGE**

```
bam2fastq.pl -b file.bam -t pe -e map -p reads -s fastq
```

### **OPTIONS**

-b (--bam) BAM alignment file  
-t (--type) Alignment type: pe (paired ends) or single [Default: pe]  
-e (--extract) Reads to extract: map, unmap [Default: map]  
-p (--prefix) Output file(s) prefix [Default: reads]  
-s (--suffix) Output file(s) suffix [Default: fastq]

## **sort\_stats.pl**      **Table generator**

This script generates tab-delimited (TSV) or comma-separated (CSV) tables from the statistics files (\*.stats) generated by get\_SNPs.pl. Single metrics tables (read mapping, total variants, variant per kb and weighted variants) can be plotted with MashR\_plotter.pl. The weighted columns correspond to: (variants found\*100)/% of reads mapping against the reference. This simple weighting underestimates the number of variants, as the reads not mapping are the ones that are the most different from the reference.

### **USAGE**

```
sort_stats.pl -s *.stats -o output_name -f csv
```

### **OPTIONS**

-h (--help) Display this list of options  
-v (--version) Display script version  
-s (--stats) Stats files obtained from get\_SNPs.pl  
-o (--output) Desired table output name; extension will be appended  
-f (--format) Desired table format, tsv or csv [Default: tsv]

## **synonymy.pl**      **Sorts synonymous/non-synonymous SNPs**

This script sorts point mutations per type (CDS, tRNA, rRNA or intergenic). If present in a CDS, the script can also differentiate between synonymous or non-synonymous, if applicable. Requires VCF files, a reference genome, and its annotation in gff format. NOTE: Genes with introns are not yet supported.

### **USAGE**

```
synonymy.pl -gcode 1 -fa reference.fasta -gff reference.gff -vcf *.vcf -o table_name
```

### **OPTIONS**

-h (--help) Display this list of options  
-v (--version) Display script version  
-fa (--fasta) Reference genome in fasta format  
-gff (--gff) Reference genome annotation in GFF format  
-vcf (--vcf) SNPs in Variant Calling Format (VCF)  
-o (--output) Table prefix [Default: synonymy] The .features/.intergenic suffixes and .tsv file extension will be added automatically.  
-gc (--gcode) NCBI genetic code; e.g.:



- 1 - The Standard Code
- 2 - The Vertebrate Mitochondrial Code
- 3 - The Yeast Mitochondrial Code
- 4 - The Mold, Protozoan, Coelenterate Mitochondrial Code and Mycoplasma/Spiroplasma Code
- 11 - The Bacterial, Archaeal and Plant Plastid Code

NOTE - For complete list; see <https://www.ncbi.nlm.nih.gov/Taxonomy/Utils/wprintgc.cgi>

### **count\_SNPs.pl**      **Quick metrics**

This script counts and summarizes the number of variants found in the user-specified VCF files. Very limited in scope.

#### **USAGE**

count\_SNPs.pl -o OUTPUT\_PREFIX -v \*.vcf

#### **OPTIONS**

-o (--output)      Desired output file prefix **## File extension (.tsv) will be appended automatically**  
 -v (--vcf)          VCF files to summarize

### **run\_Mash.pl**      **Perl wrapper for MASH**

This script runs MASH on a specified set of fasta files.

#### **USAGE**

perl run\_Mash.pl [options]

#### **EXAMPLE**

run\_Mash.pl --fasta \*.fasta --out Mash.txt --sort

#### **OPTIONS**

-f (--fasta)          Reference genome(s) in fasta file  
 -o (--out)            Output file name  
 -s (--sort)          Sort Mash output by decreasing order of similarity

### **MashToDistanceMatrix.pl**      **Generates distance matrices from MASH outputs**

This script converts the table output from MASH into a distance matrix suitable for downstream analyses.

#### **USAGE**

MashToDistanceMatrix.pl -i Mash.txt -o Mash -t csv

#### **OPTIONS**

-i (--input)          Mash output file [Default: Mash.txt]  
 -o (--out)            Output file prefix [Default: Mash]  
 -f (--format)        Output format; csv or tsv [Default: csv]

## MashR\_plotter.pl      Plots distance matrices

This script plots distances matrices created with MashToDistanceMatrix.pl. Users can specify the type (cluster, phylogenetic tree, or heatmap) and various options defined below. Clusters plots are generated using dimensionality reduction techniques. Phylogenetic trees are restricted to distance methods. Requires R.

### USAGE

MashR\_plotter.pl [OPTIONS]

### EXAMPLE (simple)

MashR\_plotter.pl -i Mash.csv -o image\_01

### EXAMPLE (advanced)

MashR\_plotter.pl -type cluster -m tsne -i Mash.csv -if csv -R Mash.R -t pdf -o image\_01 -pe 30 -it 500

### OPTIONS

--help (-h)	Display this list of options
--type (-t)	Plot type: cluster, tree, heatmap [Default: cluster]
--method (-m)	Dimensionality reduction method for clusters (mds, tsne) [Default: mds]
--input (-i)	Input file [Default: Mash.csv]
--ifformat (-if)	Input format; tsv or csv [Default: csv]
--rscript (-R)	R script output name generated [Default: Mash.R]
--format (-f)	Image format (pdf, svg, jpeg, png) [Default: pdf]
--output (-o)	Output plot name [Default: plot]
--resolution (-res)	Resolution (in DPI) [Default: 300]
--labels (-lb)	Displays labels

### ## Clustering options

--cluster (-cl)	Clustering method (pam, fanny, kmeans, clara) [Default: pam]
--nclust (-nc)	Number of clusters desired [Default: 10]

### ## t-SNE options

--perplexity (-pe)	Perplexity [Default: 30]
--iterations (-it)	Maximum number of iterations [Default: 500]
--dimensions (-di)	Number of dimensions [Default: 2]
--cmode (cm)	Color mode: rainbow, heat, terrain, topo, cm or none [Default: rainbow]

### ## Phylogenetic tree options

--treetype (-tt)	Tree type: phylogram, cladogram, fan, unrooted or radial [Default: phylogram]
--distmeth (-dm)	Distance method: nj (neighbor-joining) or upgma [Default: nj]
--outgroup (-og)	Desired outgroup from the distance matrix (e.g. -og outgroup_name)
--newick (-nw)	Phylogenetic tree output in Newick format [Default: tree.tre]

### ## Heatmap options

--colors	Desired library(RColorBrewer) colors in order of decreasing similarity [Default: white yellow red]
----------	--

--shades	Number of color shades desired [Default: 300]
--separator (-sep)	Switch off row/column separators
--clcol	Switch off column clustering
--clrow	Switch off row clustering

### ## R plotter options

--width (-wd)	Plot width [Default: 16]
--height (-he)	Plot height [Default: 10]
--fonts	Fonts [Default: Times]
--fontsize (-fs)	Plot font size [Default: 16]
--symbol (-pch)	R Plot PCH Symbols (for cluster graphs) [Default: 1]
--edges (-ed)	Draw edges (for cluster graphs)
--xrange (-x)	X-axis width (for cluster graphs) [Default: 0.005]

## REFERENCES

1. Ondov, B.D., Treangen, T.J., Mallonee, A.B., Bergman, N.H., Koren, S., and Phillippy, A.M. (2016). Fast genome and metagenome distance estimation using MinHash. *Genome Biol.* 17, 132.
2. Li, H., and Durbin, R. (2010). Fast and accurate long-read alignment with Burrows-Wheeler transform. *Bioinformatics* 26, 589–595.
3. Langmead, B., and Salzberg, S.L. (2012). Fast gapped-read alignment with Bowtie 2. *Nat Methods* 9, 357–359.
4. Kim, D., Langmead, B., and Salzberg, S.L. (2015). HISAT: a fast spliced aligner with low memory requirements. *Nat. Methods* 12, 357–360.
5. Li, H. (2018). Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics* 34, 3094–3100.
6. Sedlazeck, F.J., Rescheneder, P., Smolka, M., Fang, H., Nattestad, M., von Haeseler, A., and Schatz, M.C. (2018). Accurate detection of complex structural variations using single-molecule sequencing. *Nat. Methods* 15, 461–468.
7. Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G., and Durbin, R. (2009). The Sequence Alignment/Map format and SAMtools. *Bioinformatics* 25, 2078–2079.
8. Koboldt, D.C., Zhang, Q., Larson, D.E., Shen, D., McLellan, M.D., Lin, L., Miller, C.A., Mardis, E.R., Ding, L., and Wilson, R.K. (2012). VarScan 2: somatic mutation and copy number alteration discovery in cancer by exome sequencing. *Genome Res.* 22, 568–576.
9. Narasimhan, V., Danecek, P., Scally, A., Xue, Y., Tyler-Smith, C., and Durbin, R. (2016). BCFtools/RoH: a hidden Markov model approach for detecting autozygosity from next-generation sequencing data. *Bioinformatics* 32, 1749–1751.
10. Garrison, E., and Marth, G. (2012). Haplotype-based variant detection from short-read sequencing. *arXiv Prepr. arXiv1207.3907*, 9.
11. R Core Team (2019). R: A language and environment for statistical computing. Available at: <https://www.r-project.org/>.
12. van der Maaten, L. (2013). Barnes-Hut-SNE. *arxiv.org*, 1301.3342v2.
13. van der Maaten, L.J.P., and Hinton, G.E. (2008). Visualizing High-Dimensional Data Using t-SNE. *J. Mach. Learn. Res.* 9, 2579–2605.